

AO-A191 159

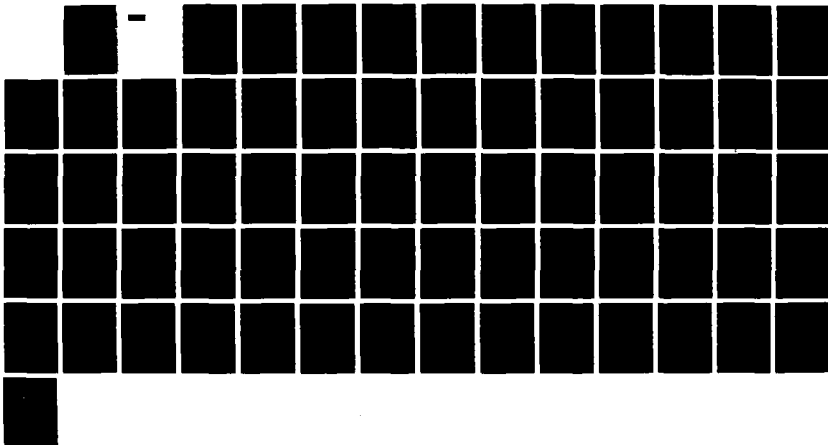
WAVFLD: A PROGRAM TO COMPUTE IONOSPHERIC HEIGHT GAIN
FUNCTIONS AND FIELD STRENGTHS AT VLF(U) NAVAL OCEAN
SYSTEMS CENTER SAN DIEGO CA J A FERGUSON ET AL. NOV 87
NOSC/TD-1192

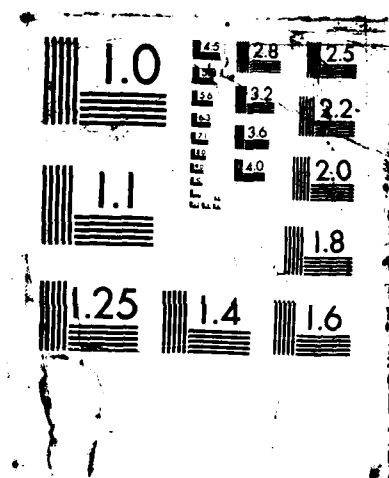
1/1

UNCLASSIFIED

F/G 20/14

NL





UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS AD-A191159	
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NOSC TD 1192			5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center	6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION		
6c. ADDRESS (City, State and ZIP Code) San Diego, CA 92152-5000	7b. ADDRESS (City, State and ZIP Code)			
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Defense Nuclear Agency (DNA) Radiation Directorate	8b. OFFICE SYMBOL (if applicable) DNA-RAAE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS (City, State and ZIP Code) Hybla Valley Federal Building Washington, DC 20305		10. SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO. 62715H	PROJECT NO. S99QMXBB	TASK NO. 544-MP20
		AGENCY ACCESSION NO. DN651-524		
11. TITLE (Include Security Classification) WAVFLD: A Program to Compute Ionospheric Height Gain Functions and Field Strengths at VLF				
12. PERSONAL AUTHOR(S) J.A. Ferguson and L.R. Hitney				
13a. TYPE OF REPORT Interim	13b. TIME COVERED FROM Oct 1986 TO Sep 1987	14. DATE OF REPORT (Year, Month, Day) November 1987	15. PAGE COUNT 71	
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SJB-GROUP		
			Atmosphere modeling Ionospheric profile	
			Radio propagation Very low frequency	
			Extremely low frequency	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) Implementation of a full-wave fields program developed for calculations at ELF is described. The program incorporates modifications to the original code for use at VLF, including allowance for multiple modes. Other changes relate to improving compatibility of the basic program setup with that of other programs in the Defense Nuclear Agency repertoire.				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL J.A. Ferguson			22b. TELEPHONE (Include Area Code) (619) 553-3062	22c. OFFICE SYMBOL Code 544

DD FORM 1473, 84 JAN

83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETEUNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

CONTENTS

Introduction	1
Description of Input	3
Program Control	3
Namelist Variables	5
Sample Problem	9
References	21

APPENDIX A

Source listing for program WAVFLD	A-1
---	-----

APPENDIX B

Source listing for program WFPLTS	B-1
A brief description of usage of program WFPLTS	B-2

ILLUSTRATIONS

1. Output from MODEFNDR program	10
2. Sample input to WAVFLD program	11
3. Printed output from WAVFLD program	12
4. Plot of the magnetoionic terms X, Y and Z for the test case	18
5. Total electric field components for the test case	18
6. Total magnetic field components for the test case	19
7. Relative Joule heating and absorption for the test case	19
8. Components of the Poynting vector for the test case	20

TABLES

1. Summary of Control Strings	3
2. A brief summary of NAMELIST variables	5

INTRODUCTION

This report describes the implementation of a full-wave fields program developed for calculations at ELF (Pappert and Shockey, 1977). The program to be described here incorporates modifications to the original code for use at VLF, including allowance for multiple modes. Other changes relate to improving compatibility of the basic program setup with that of other programs in the Defense Nuclear Agency repertoire.

The program is written in VAX FORTRAN for use with Digital Equipment Corporation's VAX/VMS operating systems. VAX FORTRAN is based on American National Standard FORTRAN-77 (ANSI X3.9-1978) and includes support for programs that conform to the previous standard (ANSI X3.9-1966) as well as numerous extensions to the ANSI standard. The WAVFLD program uses some extensions of the ANSI standard, including namelist-directed input/output and additional data typing, such as COMPLEX*16.



Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

DESCRIPTION OF INPUT

PROGRAM CONTROL

Program execution is controlled by a series of character strings which indicate the type of data to follow. The control strings must begin in column one of the record and can be in uppercase or lowercase letters. The ionospheric and collision frequency profiles can be input by either of two different methods. If either profile is strictly exponential, it can be read in by means of variables in the NAMELIST input signaled by the control string NAME. If either profile is non-exponential, it must be read in by means of a tabular-type formatted input signaled by the PROFILE or COLFREQ control strings. All of these control strings are described below, and are summarized in Table 1.

Table 1. Summary of control strings.

NAME	initiates reading of NAMELIST input
PROFILE	initiates reading of ionospheric profile
COLFREQ	initiates reading of ionospheric collision-frequency profile
DATA	initiates reading of propagation path data
SW XMT	provides same function as DATA
QUIT	indicates end of the input

NAME Signals that NAMELIST data follow. In this program the NAMELIST name is DATUM. The NAMELIST variables are described in the next section.

PROFILE Initiates reading of the ionospheric charged-particle profile data used to model the upper boundary of the earth-ionosphere waveguide. This allows for using a non-exponential ionospheric profile. The PROFILE control string is followed by an alphanumeric record, which is used to identify the profile. It can contain up to 80 characters of information. The profile is input starting at the top of the ionosphere, and the records must be input in descending order of height. The profile records contain the height, in kilometers, and the species densities, in particles per cubic centimeter, at that height. A format of [F7.2,4X,5(1X,E9.2)] is used to read the records. In the integration of the reflection elements through the ionosphere, the program interpolates exponentially between input values. The profile should contain sufficient data to define the ionospheric structure with height. For example, an exponential profile should consist of only the top and bottom heights and densities. Many regularly spaced heights tend to slow the integration. A purely exponential ionospheric profile (electrons only) may also be specified in the NAMELIST input by the variables BETA, HPRIME, and SCLHTS. A maximum of 101 heights can be specified and a maximum of five species per height may be input. The number of species to be used is determined from

the control string. If the ninth column is blank, one species is assumed. If a value is in this column, it is used for the number of species. In the special case of three species, only two species are specified at each altitude. The first is assumed to be electrons and the second positive ions. The third species, negative ions, is calculated in the program by subtracting the electron density from the positive-ion density to preserve charge neutrality. All three species are listed in the output. If the value of any species is less than 1.0×10^{-20} , it is set to that value. The end of the profile is indicated by a height less than zero.

COLFREQ Initiates reading of the collision-frequency profile via tabular input. This allows for use of a non-exponential collision frequency. If the tabular input is used, it overrides the NAMELIST variables COEFNU and EXPNU. Collision frequencies for all species must be input. The ionospheric profile must precede the collision-frequency profile because the number of species to be used is determined from the PROFILE control string. As with the ionospheric species profiles, the profile is input starting at the top of the ionosphere, the records must be input in descending order of height, and the program interpolates exponentially between input values. There need be no correspondence between the altitudes used to define the charged-particle profile and the collision-frequency profile. Also, any species value less than 1.0×10^{-20} is replaced by that value, the same format is used to read the records, and the end of the profile is indicated by a height less than zero. The profile records contain the height in kilometers and the collision frequencies in collisions per second. A maximum of 25 heights can be specified, and a maximum of 5 species per height may be input. A strictly exponential collision frequency may be specified in the NAMELIST input by the variables COEFNU and EXPNU.

DATA Signals that the propagation-path data follow. The format of these data is that which is produced by MODEFNDR (Shellman, 1986) and the SEGMENTED WAVEGUIDE PROGRAM (Ferguson and Snyder, 1987). The first record contains the data set identification. This is followed by sets of mode constants, one for each path segment. The first record for each segment contains the starting distance, frequency, magnetic azimuth, codip, and intensity, and the ground conductivity and permittivity. This record is followed by the mode-constant records, one pair for each mode, containing the following information:

```

1 THETA I T1 T2
2 THETA I T3 T4,
```

where 1 and 2 are sequencing indices and THETA is the complex eigenangle at the ground. For a discussion of I and the four T quantities, see Ferguson and Snyder, 1980. The list of modes for each segment is terminated by a

blank record. After reading the data for each segment, the program begins the WAVFLD calculations. The list of segments is terminated by a record with the starting distance set equal to 40.0.

SW XMT Provides the same function as DATA control string.

QUIT Indicates the end of the input. The execution of the program is terminated.

NAMelist VARIABLES

The NAMelist variables are described in tabular form below. The initial or default values are given in Table 2.

Table 2. Namelist variables and initial values.

Name	Value	Units
nprof	0	-
beta	0.0	1/km
hprime	0.0	km
sclhts	5.0	
h	0.0	km
alpha	3.14e-4	1/km
prec	3.0e-5	
debug	0	-
nelect	0	-
coefnu	1.816e11,4*4.54e9	coll/sec
expnu	5*-0.15	1/km
charge	-1.0,1.0,-1.0,1.0,-1.0	
mratio	1.0,4*5.8e4	
topht	100.0	km
lwstht	0.0	km
itr	0	-
maxitr	10	-
dtheta	(5.0d-2,1.0d-2)	degrees
lub	(5.0d-2,5.0d-3)	degrees
nprint	1	-
savplt	.true.	-

NPROF	A flag controlling usage of exponential electron-density profile. Setting the flag non-zero indicates usage of exponential electron-density profile. Setting the flag to zero indicates non-usage of exponential electron-density profile.
BETA	The height variation of the electron density for exponential ionospheric profiles in inverse kilometers.
HPRIME	The reference height of the electron density for exponential ionospheric profiles in kilometers.
SCLHTS	The number of scale heights above HPRIME at which the top of the exponential electron-density profile is defined.
H	Altitude in kilometers at which the modified refractive index is unity.
ALPHA	Earth curvature coefficient in inverse kilometers. ALPHA is defined as $2/\text{radius of the earth}$. For a flat earth, use ALPHA = 0.
PREC	Accuracy to be maintained locally in the numerical integrations. It is usually taken to be the default value of $3.0\text{e-}5$.
DEBUG	A flag controlling the amount of diagnostic output desired. DEBUG = 0 suppress all printout DEBUG = 1 print height gains DEBUG = 2 print information on initial boundary conditions and number of integration steps DEBUG = 3 print extensive debug output
COEFNU	The collision frequency, in collisions per second, at the ground. It is used with EXPNU to specify an exponential collision frequency.
EXPNU	The height variation in inverse kilometers. It is used to specify an exponential collision frequency.
CHARGE	The charge of each species. For an electron the charge is -1.
MRATIO	The ratio of the mass of the species to the mass of an electron. For an electron, the ratio is 1.
TOPHT	Greatest height in the ionosphere where field strengths are desired.
LWSTHT	Lowest height in the ionosphere where field strengths are desired.

ITR	A flag to control iterations. If ITR = 1, iterations are performed to refine the input eigenangle (THETA).
MAXITR	The maximum number of iterations allowed in finding a modal solution. The iterations are terminated for the current angle when MAXITR is exceeded.
DTHETA	The complex incremental change in the modal solution for computing the derivative. It is input as a pair of real numbers.
LUB	A complex number used to terminate the iterative process. The real and imaginary parts of LUB are used to test the iterative change in the real and imaginary parts of the modal solution. The iteration is stopped when the change is less than or equal to LUB in both real and imaginary parts. It is input as a pair of real numbers.
NPRINT	A flag controlling the amount of output. NPRINT = 0 no output is generated. NPRINT = 1 the total fields are printed. NPRINT = 2 the individual modes are printed.
SAVPLT	A logical flag for writing data to logical unit 10 for further processing and/or plotting. This variable allows the user to submit batch or background jobs to generate the data to be plotted. SAVPLT = .TRUE. data are written SAVPLT = .FALSE. no data are written

SAMPLE PROBLEM

A sample case will be discussed. The data for this case were generated using MODEFNDR (Shellman, 1986). The printed output for this MODEFNDR run is shown in Figure 1. This figure is included here so the user may see the correspondence between inputs to MODEFNDR and the inputs to WAVFLD (shown in Figure 2). The profile specifications are identical in the two programs. The data lines between 'DATA' and the blank line just before 'R 40' are all output from MODEFNDR. Immediately after the DATA control string is the data identification string which followed the ID string in Figure 1. The frequency, geomagnetic field parameters, and ground conductivity parameters are encoded in the line after the identification string. This is followed by pairs of lines containing the mode parameters.

The printed output from WAVFLD is shown in Figure 3. This output shows the ionospheric profile data and the values of the NAMELIST variables. After the DATA control string, the parameters of the individual modes are shown. Initially, the basic parameters of attenuation rate, normalized phase velocity, and excitation factor are printed. The number of integration steps for one integration and the resulting value of the complex mode equation are printed. The sample case calls for iteration of the input eigenangles, so the integrations continue until the incremental change in the eigenangle is less than LUB. For the first mode, this happens after two iterations. In the same fashion the remaining modes are processed. The electric and magnetic components for each mode are adjusted to correspond to the calculated excitation factor and the total fields are computed by summation. The resulting values of the fields as a function of altitude are printed. In addition, the relative Joule heating in watts per unit volume and the components of the Poynting vector are printed under the headings 'q', 'sx', 'sy' and 'sz'. The angles that the Poynting vector makes with respect to the vertical are shown under the headings 'alpha' and 'beta.' Representative plots for these test data are shown in Figures 4 through 8. It should be noted that the user may obtain other parameters (such as described by Pappert and Shockey, 1977) by modifications to the main routine.

```

PROFILE
TEST CASE
 120.00      5.80E 02
 112.00      1.10E 03
 110.00      1.30E 03
 106.00      1.70E 03
 104.00      1.90E 03
 102.00      1.98E 03
 100.00      2.00E 03
  99.00      1.95E 03
   0.00      1.83E-12
-99.
COLFREQ
 120.00      1.00E 04
 104.00      3.00E 04
   0.00      1.82E 11
-99.
ID
WAVFLD test data
NAME
&DATUM
FREQ=17.8,
AZIM=87.801,CODIP=6.593,MAGFLD=.5222D-4,
SIGMA=1.0D-5,EPSR=5.0,
RANGER=80,90,RANGEI=0,-5,
&END
THE TOP OF THE PROFILE IS SET TO BE--
 97.11      1.01E+03
AT THE TOP OF THE PROFILE B =   3.519E+01 OMEGA-R =   3.806E+07
THE BOTTOM PROFILE HEIGHT IS =   67.047 AT B(CUTOFF) =   1.000E-04
OMEGA-R EQUALS 2.5E5 AT HT =   87.06
OMEGA-R = 2.503E+05 AT HOFWR =   87.06
 83.696 -0.665   84.804 -0.101
 74.409 -7.662
 74.409 -7.662

 74.409 -7.662 USED      74.409 -7.662 DELETED
R  0.00 F 17.8000 A 87.801 C 6.593 M 0.522E-04 S 1.000E-05 E 5.0 T 87.1 H 5
MODE      THETA      ATTEN      VOVERC      WAIT MAG      WAIT ANG      THETAP
 1  84.804 -0.101   0.523   0.99624   -64.578    2.686    89.892 -4.915
 2  83.696 -0.665   4.164   0.99812   -23.736    2.850    88.836 -3.622
 3  74.409 -7.662  117.736   1.02091    16.911   -1.104    75.698 -8.396

QUIT
***THE CALCULATIONS ARE COMPLETE***
FORTRAN STOP

```

Figure 1. Sample output from MODEFNDR. The data obtained in this case are used in the sample run for WAVFLD.

```

PROFILE
TEST CASE
 120.00      5.80E 02
 112.00      1.10E 03
 110.00      1.30E 03
 106.00      1.70E 03
 104.00      1.90E 03
 102.00      1.98E 03
 100.00      2.00E 03
  99.00      1.95E 03
   0.00      1.83E-12
-99.
COLFREQ
 120.00      1.00E 04
 104.00      3.00E 04
   0.00      1.82E 11
-99.
NAME
&DATUM
TOPHT=120.0,LWSTHT=0.0,
ITR=1,
&END
DATA
WAVFLD test data
R  0.000 F 17.8000 A  87.801 C   6.593 M 0.522E-04 S 1.000E-05 E  5.0 T 87.1
1 89.89229 -4.914532 9.64114224E-06 1.96815890E-05-6.95767994E-06 2.03831610E-06
2 89.89229 -4.914532 5.02810508E-06-1.14897985E-05 1.01010334E+00 5.58272935E-03
1 88.83618 -3.622151 6.90992048E-04 2.32309173E-03-2.42370220E-07-1.95802787E-08
2 88.83618 -3.622151-1.51578261E-05 1.88085523E-05 1.01016688E+00 5.75073995E-03
1 75.69807 -8.396441-2.33791590E-01-1.40563220E-01-5.26763112E-11 4.32070248E-11
2 75.69807 -8.396441-4.27416853E-06 3.35867099E-07 1.01005256E+00 8.58251192E-03

R 40.

```

Figure 2. Sample input to the WAVFLD program.

```

PROFILE
TEST CASE
120.00 5.80E+02
112.00 1.10E+03
110.00 1.30E+03
106.00 1.70E+03
104.00 1.90E+03
102.00 1.98E+03
100.00 2.00E+03
99.00 1.95E+03
0.00 1.83E-12
COLFREQ
120.00 1.00E+04
104.00 3.00E+04
0.00 1.82E+11
NAME
STATUS
= 0.0000000000000000E+00,
= 3.1400000000000000E-04,
= 3.0000000000000000E-05,
= 0,
= 0,
= 181600000000.0000 , 4*5400000000.000000 ,
COEFNU = 5*-0.1500000000000000 , 1.0000000000000000 ,
EXNU = -1.0000000000000000 , 1.0000000000000000 ,
CHARGE = -1.0000000000000000 , 1.0000000000000000 ,
-1.0000000000000000 , 1.0000000000000000 , 4*58000.000000000000 ,
NRATIO = 1.0000000000000000 , 4*58000.000000000000 ,
NPROF = 0,
BETA = 0.0000000000000000E+00,
HPRIME = 0.0000000000000000E+00,
SCIHS = 5.0000000000000000 ,
TOHIT = 120.00000000000000 ,
LWSHIT = 0.0000000000000000E+00,
ITR = 1,
MAXTR = 10,
DIHEPA = 5.0000000000000000E-02,
LUB = 5.0000000000000000E-02,
NPRINT = 1,
SAVPLT = 1,
= T
DATA
mode real imag atten v/c wait's exc real' imag'
1 89.89229 -4.91453 0.523 0.99633 -64.260 2.686 89.89229 -4.91453
188 integration steps used in wavfld
modal eqn value= -3.37864E-01 -2.35654E-01
modal eqn value= -2.98840E-01 5.59053E-02
new theta= 89.900 -4.967
188 integration steps used in wavfld
modal eqn value= -2.50051E-02 -3.20914E-01
modal eqn value= 2.80062E-02 1.31133E-02
new theta= 89.897 -4.963
188 integration steps used in wavfld
modal eqn value= 7.89829E-04 -1.78629E-03
mode real imag atten v/c wait's exc real' imag'
2 88.83618 -3.62215 4.164 0.99821 -23.418 2.850 88.83618 -3.62215

```

Figure 3. Printed output from WAVFLD program.

```

188 integration steps used in wavfld
modal eqn value= 2.14061E-01 -9.11512E-02
modal eqn value= 1.00186E-01 -1.27949E-01
new theta= 88.872 -3.681
188 integration steps used in wavfld
modal eqn value= 1.03976E-01 3.90162E-02
modal eqn value= -7.03339E-03 -1.19096E-02
new theta= 88.868 -3.685
188 integration steps used in wavfld
modal eqn value= -4.81178E-04 -2.35637E-04
mode real imag atten v/c
3 75.69807 -8.39644 117.727 1.02100
188 integration steps used in wavfld
modal eqn value= 1.07031E+02 5.21118E+01
modal eqn value= 1.80293E+02 3.14973E+01
new theta= 75.600 -8.470
188 integration steps used in wavfld
modal eqn value= -8.10557E+01 2.18985E+01
modal eqn value= -3.53444E+00 -9.17321E-01
new theta= 75.602 -8.469
188 integration steps used in wavfld
modal eqn value= -4.59039E-02 1.72921E-02

```

WAVFLD test data

r 0.000 f 17.8000 a 87.801 c 6.593 m 0.522 s 1.000E-05 e 5.0 h 0.0

mag(y) 8.209E+01 modes 3

total Field strengths computed by WAVFLD

ht	mag	x	mag	y	mag	z	ang	q	SX	SY	SZ	alpha	beta
120.0	e	2.068E-03	0.483	1.556E-03	2.002	1.844E-04	1.832	3.531E-07	2.294E-06	-7.562E-07	1.043E-05	77.597	94.146
118.8	h	2.484E-03	-1.145	3.177E-03	0.441	1.543E-03	1.958	3.938E-07	2.312E-06	-7.113E-07	1.035E-05	77.410	93.931
117.6	h	2.504E-03	-0.421	3.168E-03	1.169	1.549E-03	2.673	4.322E-07	2.235E-06	-6.876E-07	1.027E-05	77.726	93.830
116.4	h	2.587E-03	0.339	3.227E-03	1.923	1.527E-03	-2.879	4.694E-07	2.146E-06	-6.837E-07	1.020E-05	78.118	93.835
115.2	e	1.860E-03	2.688	1.505E-03	-2.056	1.773E-04	-2.201	5.238E-07	2.163E-06	-6.909E-07	1.013E-05	77.943	93.903
114.0	h	2.697E-03	1.104	3.301E-03	2.682	1.493E-03	-1.101	5.948E-07	2.171E-06	-6.964E-07	1.005E-05	77.814	93.962
112.8	h	2.748E-03	1.889	3.296E-03	-2.819	1.491E-03	-1.287	6.544E-07	2.041E-06	-6.925E-07	9.987E-06	78.448	93.967
111.6	h	2.777E-03	2.730	3.273E-03	-1.979	1.496E-03	-0.468	7.173E-07	1.941E-06	-6.871E-07	9.926E-06	78.935	93.960
110.4	e	1.718E-03	-1.111	1.469E-03	0.424	1.725E-04	0.380	8.259E-07	1.963E-06	-6.868E-07	9.869E-06	78.751	93.981
109.2	h	2.886E-03	-2.675	3.348E-03	-1.101	1.457E-03	0.380	9.242E-07	1.879E-06	-6.828E-07	9.815E-06	79.162	93.979
108.0	h	1.631E-03	0.748	1.429E-03	2.283	1.669E-04	1.293	1.003E-06	1.765E-06	-6.757E-07	9.768E-06	79.759	93.957
106.8	h	3.017E-03	-0.862	3.409E-03	0.711	1.418E-03	2.239	1.153E-06	1.790E-06	-6.736E-07	9.725E-06	79.569	93.962
105.6	h	1.585E-03	1.706	1.407E-03	-3.043	1.645E-04	3.133	1.288E-06	1.712E-06	-6.683E-07	9.686E-06	79.979	93.947
104.4	e	1.513E-03	-2.495	1.364E-03	-0.961	1.587E-04	-1.066	1.393E-06	1.622E-06	-6.640E-07	9.654E-06	80.464	93.934

Figure 3. Printed output from WAVFLD program (contd).

103.2	h	3.390E-03	-1.896	3.684E-03	-0.324	1.291E-03	1.196	1.690E-06	1.668E-06	-6.656E-07	9.627E-06	80.169	93.955
	e	1.435E-03	0.865	1.316E-03	2.397	1.526E-04	2.303						
102.0	h	3.357E-03	-0.752	3.635E-03	0.819	1.306E-03	2.353	2.020E-06	1.608E-06	-6.597E-07	9.609E-06	80.499	93.927
	e	1.413E-03	2.014	1.300E-03	-2.737	1.505E-04	-2.833						
100.8	h	3.402E-03	0.428	3.673E-03	1.999	1.290E-03	-2.782	2.388E-06	1.603E-06	-6.560E-07	9.601E-06	80.521	93.909
	e	1.398E-03	3.077	1.289E-03	-1.544	1.485E-04	-1.641						
99.6	h	3.438E-03	1.579	3.702E-03	3.129	1.280E-03	-1.589	2.937E-06	1.672E-06	-6.744E-07	9.603E-06	80.123	94.018
	e	1.417E-03	-1.906	1.321E-03	-0.380	1.518E-04	-0.477						
98.4	h	3.361E-03	2.755	3.647E-03	-1.944	1.311E-03	-0.425	2.946E-06	1.659E-06	-7.310E-07	9.614E-06	80.207	94.348
	e	1.422E-03	-0.730	1.316E-03	0.758	1.496E-04	0.646						
97.2	h	3.315E-03	-2.411	3.711E-03	0.823	1.306E-03	0.713	3.177E-06	2.149E-06	-8.688E-07	9.616E-06	77.401	95.163
	e	1.656E-03	0.323	1.470E-03	1.773	1.633E-04	1.648						
96.0	h	2.889E-03	-1.544	3.392E-03	0.066	1.459E-03	1.728	3.548E-06	2.659E-06	-1.104E-06	9.605E-06	74.527	96.555
	e	1.923E-03	1.108	1.658E-03	2.509	1.823E-04	2.355						
94.8	h	2.368E-03	-0.746	3.024E-03	0.911	1.647E-03	2.464	3.894E-06	2.683E-06	-1.435E-06	9.591E-06	74.373	98.496
	e	2.049E-03	1.787	1.704E-03	3.091	1.836E-04	2.859						
93.6	h	2.071E-03	-0.036	2.981E-03	1.665	1.694E-03	3.046	4.873E-06	2.464E-06	-1.853E-06	9.596E-06	75.598	100.932
	e	2.173E-03	2.445	1.668E-03	-2.671	1.710E-04	-3.058						
92.4	h	2.420E-03	3.047	1.640E-03	-2.193	1.584E-04	-2.864	6.890E-06	2.294E-06	-2.354E-06	9.643E-06	76.620	103.716
	e	1.651E-03	0.959	2.956E-03	2.800	1.633E-03	-2.240						
91.2	h	2.735E-03	-2.724	1.653E-03	-1.775	1.760E-04	-2.887	1.033E-05	2.227E-06	-2.924E-06	9.757E-06	77.141	106.685
	e	1.418E-03	1.319	2.788E-03	-2.959	1.646E-03	-1.825						
90.0	h	3.004E-03	-2.278	1.692E-03	1.425	2.826E-04	-2.943	1.703E-05	2.201E-06	-3.571E-06	9.987E-06	77.570	109.676
	e	1.200E-03	1.632	1.669E-03	-2.397	1.684E-03	-1.478						
88.8	h	3.194E-03	-1.856	1.755E-03	-1.138	5.600E-04	-2.864	3.280E-05	2.219E-06	-4.296E-06	1.048E-05	78.038	112.300
	e	1.015E-03	1.921	2.733E-03	-1.819	1.724E-03	-1.193						
87.6	h	3.402E-03	1.433	1.763E-03	-0.898	1.140E-03	-2.744	6.958E-05	2.745E-06	-5.049E-06	1.158E-05	76.667	113.551
	e	8.706E-04	2.203	2.944E-03	-1.270	1.748E-03	-0.955						
86.4	h	3.808E-03	-1.035	1.764E-03	-0.691	2.163E-03	-2.675	1.353E-04	5.564E-06	-5.648E-06	1.393E-05	68.225	112.072
	e	7.739E-04	2.488	3.427E-03	-0.762	1.745E-03	-0.749						
85.2	h	4.438E-03	-0.734	1.739E-03	-0.502	3.549E-03	-2.650	2.003E-04	1.340E-05	-5.866E-06	1.783E-05	53.074	108.215
	e	7.207E-04	2.755	4.136E-03	-0.290	1.716E-03	-0.560						
84.0	h	5.050E-03	-0.543	1.694E-03	-0.320	5.099E-03	-2.603	2.272E-04	2.738E-05	-5.607E-06	2.248E-05	39.389	104.003
	e	6.881E-04	2.987	5.306E-03	0.107	1.667E-03	-0.379						
82.8	h	5.449E-03	-0.429	1.640E-03	-0.142	6.837E-03	-2.524	2.240E-04	4.909E-05	-4.757E-06	2.673E-05	28.563	100.093
	e	6.583E-04	3.095	6.960E-03	0.399	1.610E-03	-0.202						
81.6	h	5.605E-03	-0.365	1.585E-03	0.033	8.791E-03	-2.443	2.030E-04	8.085E-05	-3.295E-06	2.962E-05	20.124	96.346
	e	6.263E-04	2.916	8.961E-03	0.599	1.551E-03	-0.026						
80.4	h	5.554E-03	-0.341	1.534E-03	0.205	1.090E-02	-2.374	1.711E-04	1.238E-04	-1.353E-06	3.042E-05	13.807	92.547
	e	5.911E-04	2.752	1.115E-02	0.733	1.497E-03	0.147						
79.2	h	5.350E-03	-0.353	1.489E-03	0.371	1.308E-02	-2.321	1.351E-04	1.776E-04	8.689E-07	2.852E-05	9.123	88.255
	e	5.528E-04	2.600	1.342E-02	0.821	1.450E-03	0.316						
78.0	h	5.049E-03	-0.401	1.450E-03	0.532	1.525E-02	-2.284	1.007E-04	2.409E-04	3.162E-06	2.355E-05	5.583	82.351
	e	5.122E-04	2.455	1.566E-02	0.877	1.410E-03	0.479						
76.8	h	4.705E-03	-0.487	1.417E-03	0.683	1.377E-02	-2.263	7.163E-05	3.116E-04	5.332E-06	1.532E-05	2.815	70.813
	e	4.702E-04	2.313	1.784E-02	0.908	1.377E-03	0.633						
75.6	h	4.375E-03	-0.613	1.388E-03	0.825	1.939E-02	-2.256	4.894E-05	3.875E-04	7.210E-06	3.848E-06	0.569	28.090
	e	4.282E-04	2.170	1.990E-02	0.920	1.349E-03	0.777						
74.4	h	4.116E-03	-0.780	1.361E-03	0.956	1.229E-02	-2.262	3.236E-05	4.664E-04	8.659E-06	-1.076E-05	-1.321	-51.168
	e	3.873E-04	2.023	2.184E-02	0.916	1.323E-03	0.280						
73.2	h	3.984E-03	-0.982	1.332E-03	1.077	2.306E-02	-2.280	2.083E-05	5.466E-04	9.579E-06	-2.828E-05	-2.962	-71.289
	e	3.485E-04	-1.868	2.364E-02	0.899	1.297E-03	1.032						
72.0	h	4.019E-03	-1.202	1.299E-03	1.186	2.471E-02	-2.310	1.313E-05	6.269E-04	9.910E-06	-4.845E-05	-4.420	-78.440
	e	3.129E-04	-1.702	2.532E-02	0.870	1.267E-03	1.143						

Figure 3. Printed output from WAVFLD program (contd.).

70.8	e	4.230E-03	-1.419	1.261E-03	1.286	2.625E-02	-2.349	8.136E-06	7.066E-04	9.637E-06	-7.095E-05	-5.733	-82.265
69.6	h	4.815E-04	-1.524	2.687E-02	0.830	1.232E-03	-1.243	4.978E-06	7.860E-04	8.785E-06	-9.544E-05	-6.924	-84.741
68.4	h	2.548E-04	-1.331	2.834E-02	0.780	1.191E-03	-1.333	3.019E-06	8.656E-04	7.421E-06	-1.216E-04	-7.997	-86.508
67.2	h	2.080E-03	-1.788	1.165E-03	1.456	2.908E-02	-2.456	1.821E-06	9.466E-04	5.648E-06	-1.491E-04	-8.952	-87.831
66.0	e	5.640E-03	-1.935	1.106E-03	1.528	1.143E-03	-1.414	1.094E-06	1.031E-03	3.597E-06	-1.777E-04	-9.784	-88.841
64.8	h	2.178E-04	-0.912	3.110E-02	0.654	1.088E-03	-1.485	6.569E-07	1.120E-03	1.420E-06	-2.073E-04	-10.485	-89.607
63.6	h	6.240E-03	-2.063	1.039E-03	1.591	3.175E-02	-2.593	3.941E-07	1.216E-03	-7.195E-07	-2.376E-04	-11.055	-90.174
62.4	e	2.073E-04	-0.696	3.245E-02	0.580	1.026E-03	-1.547	2.365E-07	1.321E-03	-2.661E-06	-2.687E-04	-11.496	-90.567
61.2	h	6.856E-03	-2.177	9.658E-04	1.645	3.309E-02	-2.672	1.419E-07	1.437E-03	-4.255E-06	-3.006E-04	-11.819	-90.811
60.0	h	2.014E-04	-0.484	3.383E-02	0.500	9.563E-04	-1.600	8.512E-08	1.564E-03	-5.375E-06	-3.336E-04	-12.041	-90.923
58.8	e	7.466E-03	-2.282	8.864E-04	1.689	3.448E-02	-2.755	5.103E-08	1.703E-03	-5.923E-06	-3.678E-04	-12.185	-90.922
57.6	h	1.991E-04	-0.284	3.526E-02	0.415	8.810E-04	-1.643	3.055E-08	1.855E-03	-5.841E-06	-4.037E-04	-12.276	-90.829
56.4	h	8.058E-03	-2.381	8.021E-04	1.723	3.592E-02	-2.843	1.885E-21	2.018E-03	-5.115E-06	-4.415E-04	-12.340	-90.664
55.2	e	8.622E-03	-2.478	7.141E-04	1.743	3.745E-02	-2.933	0.000E+00	2.192E-03	-3.775E-06	-4.819E-04	-12.400	-90.449
54.0	h	2.007E-04	-0.072	3.817E-02	0.234	7.169E-04	-1.693	2.301E-25	2.373E-03	-1.895E-06	-5.251E-04	-12.477	-90.207
52.8	h	9.154E-03	-2.574	6.239E-04	1.747	3.905E-02	-3.026	0.000E+00	2.560E-03	4.105E-07	-5.718E-04	-12.589	-89.959
51.6	h	2.026E-04	-0.226	4.005E-02	0.140	6.310E-04	-1.693	1.180E-22	2.751E-03	2.991E-06	-6.224E-04	-12.750	-89.725
50.4	h	9.654E-03	-2.672	5.337E-04	1.726	4.073E-02	-3.121	-7.539E-21	2.942E-03	5.672E-06	-6.775E-04	-12.968	-89.520
49.2	h	2.042E-04	-0.366	4.183E-02	0.044	5.451E-04	-1.670	0.000E+00	3.132E-03	8.260E-06	-7.375E-04	-13.252	-89.358
48.0	h	1.013E-02	-2.772	4.463E-04	1.670	4.247E-02	-3.067	-7.539E-21	3.318E-03	1.056E-05	-8.029E-04	-13.603	-89.247
46.8	h	2.051E-04	-0.494	4.368E-02	-0.052	4.621E-04	-1.614	0.000E+00	3.502E-03	1.238E-05	-8.745E-04	-14.021	-89.189
45.6	h	1.057E-02	-2.876	3.660E-04	1.560	4.427E-02	-2.971	4.712E-22	3.683E-03	1.354E-05	-9.526E-04	-14.503	-89.185
44.4	h	2.049E-04	-0.612	4.560E-02	-0.148	3.863E-04	-1.507	0.000E+00	3.864E-03	1.391E-05	-1.038E-03	-15.039	-89.232
43.2	h	1.101E-02	-2.985	3.004E-04	1.368	4.610E-02	-2.875	-4.712E-22	4.049E-03	1.339E-05	-1.132E-03	-15.617	-89.322
42.0	h	2.034E-04	-0.721	4.755E-02	-0.245	3.244E-04	-1.330	0.000E+00	4.245E-03	1.190E-05	-1.235E-03	-16.216	-89.448
40.8	h	1.145E-02	-3.098	2.794E-02	1.076	4.794E-02	-1.777	4.712E-22	4.461E-03	9.440E-06	-1.348E-03	-16.812	-89.599
39.6	h	2.006E-04	-0.825	4.951E-02	-0.343	2.860E-04	-1.072	-3.769E-21	4.708E-03	6.055E-06	-1.473E-03	-17.375	-89.764
38.4	h	1.190E-02	-3.067	2.580E-04	0.732	4.976E-02	-2.679	0.000E+00	5.000E-03	1.837E-06	-1.612E-03	-17.873	-89.935
		1.964E-04	0.923	5.146E-02	-0.440	5.154E-02	-2.580						
		1.238E-02	2.945	5.338E-02	0.438	5.154E-02	2.580						
		1.290E-02	2.818	3.467E-02	-0.240	5.327E-02	2.480						
		1.840E-04	1.108	5.524E-02	-0.639	3.554E-04	0.296						
		1.349E-02	2.689	4.134E-04	0.121	5.493E-02	2.377						
		1.761E-04	1.198	5.702E-02	-0.741	4.167E-04	0.169						
		1.416E-02	2.558	4.838E-04	0.052	5.651E-02	2.272						
		1.672E-04	1.287	5.873E-02	-0.846	4.830E-04	0.092						
		1.491E-02	2.426	5.538E-04	0.014	5.803E-02	2.164						
		1.575E-04	1.376	6.036E-02	-0.953	5.500E-04	0.047						
		1.575E-02	2.295	6.211E-04	-0.004	5.948E-02	2.052						
		1.470E-04	1.467	6.193E-02	-1.064	6.150E-04	0.023						
		1.670E-02	2.164	6.844E-04	-0.009	6.090E-02	1.936						
		1.361E-04	1.561	6.345E-02	-1.179	6.766E-04	0.013						
		1.774E-02	2.036	7.428E-04	-0.005	6.233E-02	1.816						
		1.250E-04	1.659	6.498E-02	-1.298	7.337E-04	0.012						
		1.889E-02	1.909	7.956E-04	0.004	6.381E-02	1.690						
		1.137E-04	1.763	6.655E-02	-1.422	7.857E-04	0.019						
		2.013E-02	1.785	8.426E-04	0.018	6.540E-02	1.560						
		1.025E-04	1.876	8.233E-02	-1.551	8.321E-04	0.030						
		2.147E-02	1.663	8.836E-04	0.035	6.719E-02	1.425						
		9.162E-05	2.002	7.009E-02	-1.685	8.727E-04	0.044						
		2.289E-02	1.542	9.183E-04	0.054	6.925E-02	1.285						

Figure 3. Printed output from WAVFLD program (contd).

37.2	h	8.137E-05	2.143	7.223E-02	-1.823	9.072E-04	0.061	0.000E+00	5.353E-03	-3.072E-06	-1.768E-03	-18.275	-90.100
	e	2.440E-02	1.424	9.468E-04	0.074	7.168E-02	1.142		5.787E-03	-8.488E-06	-1.942E-03	-18.556	-90.250
36.0	h	7.202E-05	2.306	7.472E-02	-1.964	9.358E-04	0.079	0.000E+00	6.322E-03	-1.419E-05	-2.140E-03	-18.699	-90.380
	e	2.599E-02	1.307	9.692E-04	0.095	7.455E-02	0.096	-9.423E-22	6.982E-03	-1.994E-05	-2.363E-03	-18.702	-90.483
34.8	h	6.393E-05	2.494	7.766E-02	-2.109	9.584E-04	0.099	0.000E+00	7.793E-03	-2.548E-05	-2.619E-03	-18.574	-90.558
	e	2.765E-02	1.192	9.856E-04	0.117	7.796E-02	0.088	7.538E-21	8.782E-03	-3.056E-05	-2.910E-03	-18.336	-90.602
33.6	h	5.749E-05	2.710	8.114E-02	-2.256	9.750E-04	0.120	0.000E+00	9.980E-03	-3.493E-05	-3.246E-03	-18.015	-90.617
	e	2.940E-02	1.077	9.961E-04	0.140	8.148E-02	0.070	-1.319E-20	1.142E-02	-3.837E-05	-3.631E-03	-17.641	-90.605
32.4	h	5.310E-05	2.952	8.522E-02	-2.403	9.859E-04	0.141	7.538E-21	1.313E-02	-4.070E-05	-4.075E-03	-17.241	-90.572
	e	3.122E-02	0.962	1.001E-03	0.163	8.667E-02	0.552	0.000E+00	1.515E-02	-4.177E-05	-4.586E-03	-16.840	-90.522
31.2	h	5.098E-05	-3.075	8.998E-02	-2.550	9.913E-04	0.163	1.886E-21	1.752E-02	-4.148E-05	-5.175E-03	-16.455	-90.459
	e	3.313E-02	0.848	1.001E-03	0.185	9.207E-02	0.406	-1.508E-20	2.028E-02	-3.981E-05	-5.854E-03	-16.099	-90.390
30.0	h	5.110E-05	-2.822	9.545E-02	-2.695	9.914E-04	0.184	3.039E-20	2.348E-02	-3.678E-05	-6.635E-03	-15.781	-90.318
	e	3.509E-02	-2.589	1.017E-01	-2.838	9.864E-04	0.206	-4.601E-25	2.715E-02	-3.245E-05	-7.532E-03	-15.503	-90.247
28.8	h	3.721E-02	0.619	9.846E-04	0.230	1.051E-01	0.222	0.000E+00	3.137E-02	-2.697E-05	-8.562E-03	-15.267	-90.180
	e	5.645E-05	-2.386	1.087E-01	-2.979	9.766E-04	0.228	3.015E-20	3.618E-02	-2.052E-05	-9.743E-03	-15.072	-90.121
27.6	h	3.940E-02	0.504	9.698E-04	0.252	1.128E-01	0.228	2.639E-20	4.165E-02	-1.334E-05	-1.109E-02	-14.913	-90.069
	e	4.171E-02	0.388	9.507E-04	0.274	1.212E-01	0.150	3.769E-21	4.787E-02	-5.689E-06	-1.264E-02	-14.788	-90.026
26.4	h	6.065E-05	-2.212	1.165E-01	-3.116	9.624E-04	0.249	0.000E+00	5.491E-02	2.128E-06	-1.440E-02	-14.693	-89.992
	e	4.414E-02	0.272	9.278E-04	0.295	1.304E-01	0.280	-1.508E-20	6.289E-02	9.796E-06	-1.641E-02	-14.623	-89.966
25.2	h	7.003E-05	-1.939	1.344E-01	2.903	9.217E-04	0.290	7.539E-21	7.191E-02	1.699E-05	-1.870E-02	-14.574	-89.948
	e	4.673E-02	0.156	9.014E-04	0.315	1.404E-01	0.408	3.769E-21	8.211E-02	2.340E-05	-2.130E-02	-14.543	-89.937
24.0	h	7.470E-05	-1.830	1.446E-01	2.775	8.959E-04	0.310	1.150E-25	9.364E-02	2.870E-05	-2.426E-02	-14.524	-89.932
	e	4.948E-02	0.039	8.718E-04	0.335	1.511E-01	0.532	5.751E-26	1.067E-01	3.261E-05	-2.762E-02	-14.516	-89.932
22.8	h	7.915E-05	-1.734	1.555E-01	2.650	8.669E-04	0.330	0.000E+00	1.215E-01	3.486E-05	-3.145E-02	-14.514	-89.936
	e	5.241E-02	-0.079	8.393E-04	0.354	1.626E-01	0.655	0.000E+00	1.382E-01	3.524E-05	-3.579E-02	-14.517	-89.944
21.6	h	8.332E-05	-1.649	1.671E-01	2.527	8.350E-04	0.348	2.356E-22	1.572E-01	3.356E-05	-4.073E-02	-14.521	-89.953
	e	5.554E-02	-0.197	8.042E-04	0.372	1.748E-01	0.775						
20.4	h	5.715E-05	-1.573	1.796E-01	2.407	8.004E-04	0.366						
	e	5.889E-02	-0.315	7.667E-04	0.390	1.878E-01	0.894						
19.2	h	9.062E-05	-1.504	1.928E-01	2.288	7.635E-04	0.383						
	e	6.248E-02	-0.433	7.273E-04	0.406	2.016E-01	1.011						
18.0	h	9.373E-05	-1.440	2.068E-01	2.170	7.246E-04	0.399						
	e	6.634E-02	-0.551	6.861E-04	0.421	2.161E-01	1.127						
16.8	h	6.649E-05	-1.382	2.217E-01	2.054	6.838E-04	0.413						
	e	7.048E-02	-0.669	6.433E-04	0.435	2.315E-01	1.242						
15.6	h	9.891E-05	-1.329	2.374E-01	1.939	6.414E-04	0.427						
	e	7.493E-02	-0.786	5.992E-04	0.447	2.478E-01	1.356						
14.4	h	1.010E-04	-1.279	2.540E-01	1.824	5.977E-04	0.439						
	e	7.971E-02	-0.904	5.540E-04	0.458	2.650E-01	1.470						
13.2	h	1.029E-04	-1.234	2.715E-01	1.710	5.528E-04	0.449						
	e	8.484E-02	-1.021	5.078E-04	0.467	2.833E-01	1.584						
12.0	h	1.044E-04	-1.192	2.901E-01	1.596	5.069E-04	0.458						
	e	9.034E-02	-1.137	4.609E-04	0.474	3.026E-01	1.698						
10.8	h	1.058E-04	-1.154	3.097E-01	1.482	4.602E-04	0.465						
	e	9.625E-02	-1.254	4.134E-04	0.479	3.230E-01	1.811						
9.6	h	1.069E-04	-1.119	3.305E-01	1.369	4.128E-04	0.469						
	e	1.026E-01	-1.369	3.653E-04	0.480	3.447E-01	1.924						
8.4	h	1.079E-04	-1.088	3.526E-01	1.255	3.649E-04	0.471						
	e	1.094E-01	-1.484	3.168E-04	0.478	3.678E-01	2.038						
7.2	h	1.087E-04	-1.061	3.761E-01	1.142	3.165E-04	0.468						
	e	1.167E-01	-1.599	2.681E-04	0.470	3.923E-01	2.151						
6.0	h	1.093E-04	-1.037	4.011E-01	1.028	2.679E-04	0.460						

Figure 3. Printed output from WAVFLD program (contd).

```

4.8 e 1.244E-01 -1.713 2.192E-04 0.453 4.185E-01 -2.265 -7.189E-27 1.788E-01 2.967E-05 -4.634E-02 -14.526 -89.963
h 1.099E-04 -1.018 4.276E-01 0.915 2.191E-04 0.443
3.6 e 1.328E-01 -1.826 1.703E-04 0.422 4.464E-01 -2.379 1.885E-21 2.034E-01 2.349E-05 -5.271E-02 -14.529 -89.974
h 1.103E-04 -1.002 4.560E-01 0.801 1.702E-04 0.412
2.4 e 1.417E-01 -1.939 1.216E-04 0.360 4.762E-01 -2.492 1.885E-21 2.314E-01 1.496E-05 -5.996E-02 -14.529 -89.986
h 1.107E-04 -0.990 4.862E-01 0.688 1.215E-04 0.350
1.2 e 1.512E-01 -2.052 7.403E-05 0.210 5.080E-01 -2.606 -1.123E-28 2.632E-01 4.092E-06 -6.820E-02 -14.526 -89.997
h 1.110E-04 -0.983 5.185E-01 0.574 7.400E-05 0.199
0.0 e 1.614E-01 -2.164 3.339E-05 -0.376 5.420E-01 -2.719 0.000E+00 2.996E-01 -9.070E-06 -7.758E-02 -14.519 -90.007
h 1.113E-04 -0.979 5.531E-01 0.460 3.337E-05 -0.387

```

```

FORTRAN STOP

```

Figure 3. Printed output from WAVFLD program (contd).

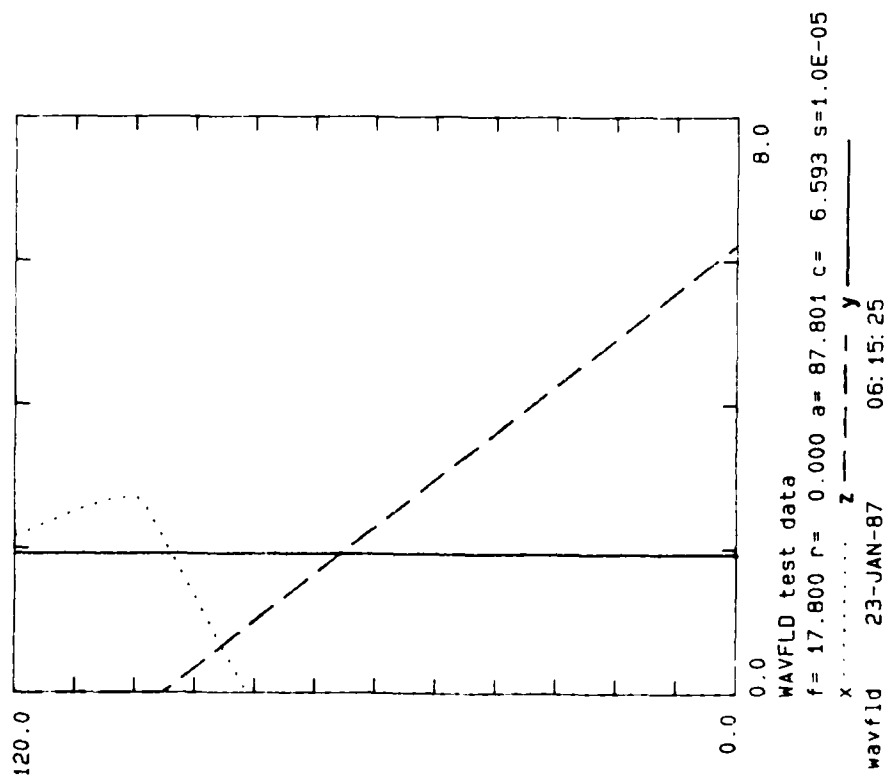


Figure 4. Plot of the magnetoionic terms X, Y and Z for the test case.

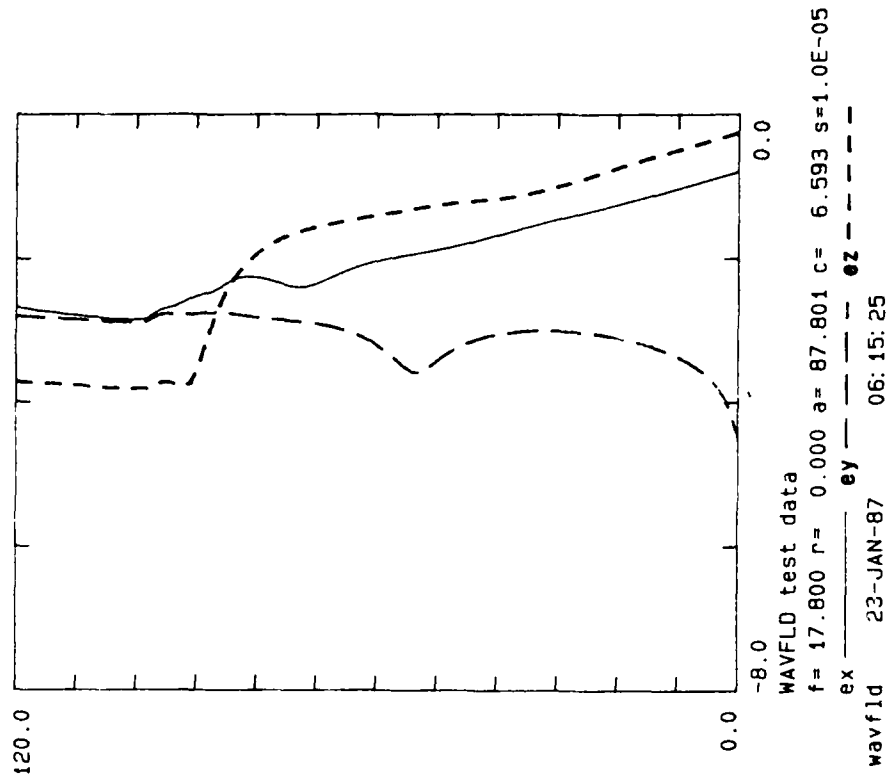


Figure 5. Total electric field components for the test case.

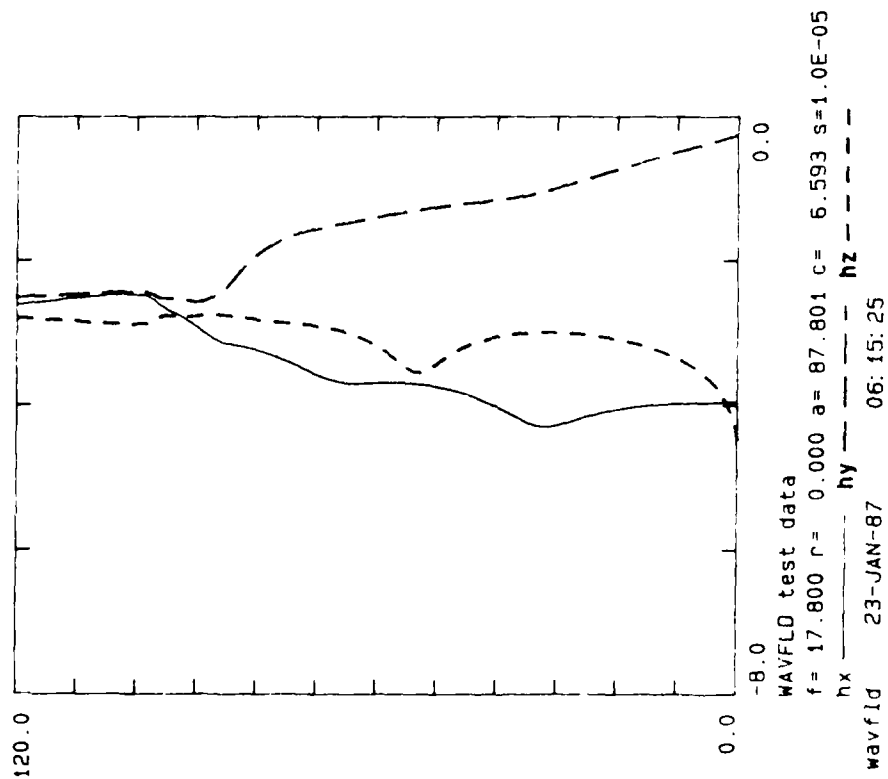


Figure 6. Total magnetic field components for the test case.

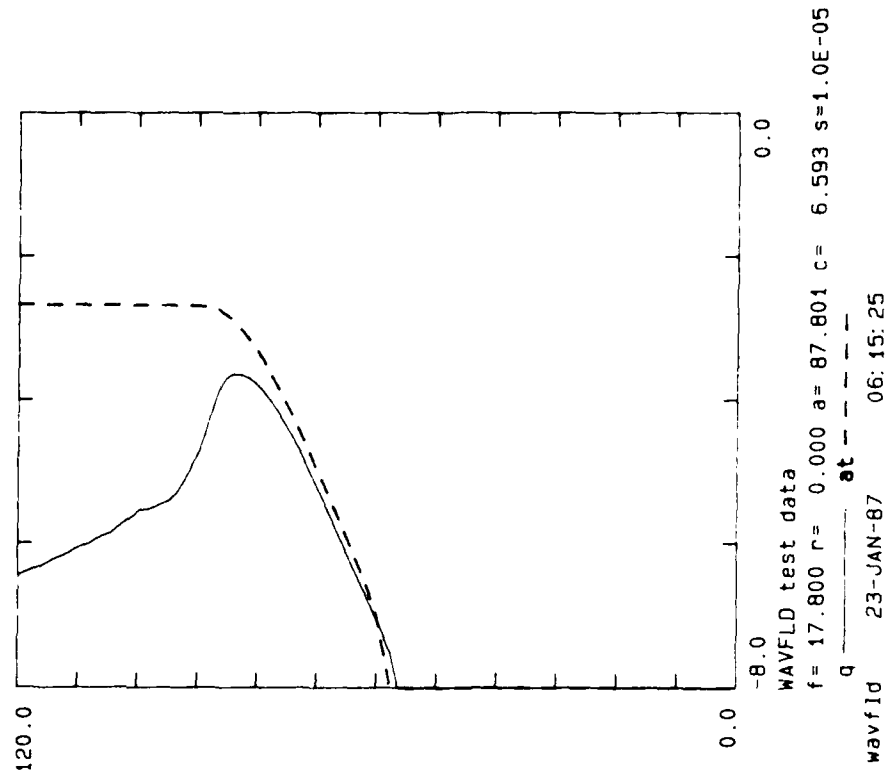


Figure 7. Relative Joule heating and absorption for the test case.

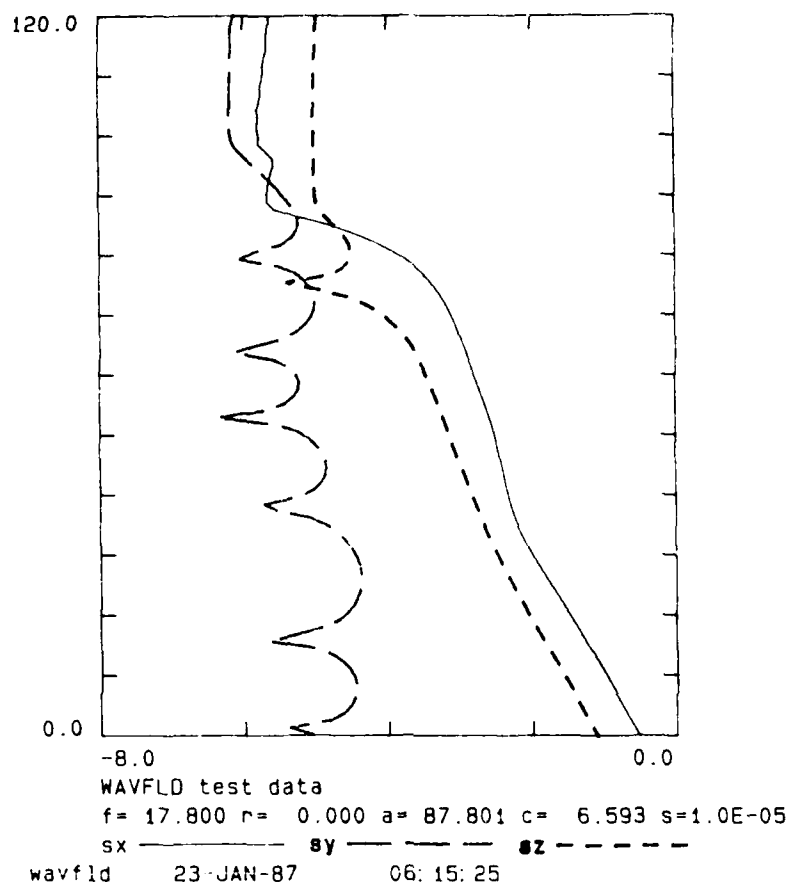


Figure 8. Components of the Poynting vector for the test case.

REFERENCES

1. Ferguson, J. A., and F. P. Snyder, "Approximate VLF/LF Waveguide Mode Conversion Model," NOSC TD 400, November 1980.
2. Ferguson, J. A., and F. P. Snyder, "The Segmented Waveguide Program for Long Wavelength Propagation Calculations," NOSC TD 1071, April 1987.
3. Pappert, R. A., and L. R. Shockey, "Ionospheric Reflection and Absorption Properties of Normal Modes at ELF," DNA Interim Report 772, September 1977.
4. Shellman, C. H., "A New Version of MODESRCH Using Interpolated Values of the Magnetoionic Reflection Coefficients," NOSC TR 1143, August 1986.

APPENDIX A

SOURCE LISTING FOR PROGRAM WAVFLD

```

c      wavfld
c
c      namelist:  nprof,beta,hprime,sclhts,
c                  h,alpha,prec,debug,nelect,
c                  coefnu,expnu,charge,mratio,
c                  topht,lwstht,itr,maxitr,dtheta,lub,
c                  nprint,savplt
c
c      nprint=0    gives no print
c                  1      print of total fields etc.
c                  2      print of individual modes
c
c      savplt=t     writes data to unit 10 for processing by wfplts
c                  =f     no output to unit 10
c
c      implicit real*8 (a-h,o-z)
c      common/wfinpt/thetap,freq,azim,codip,magfld,coefnu(5),expnu(5),
$      topht,lwstht,delht,h,alpha,sigma,epsr,nelect
c      common/wf flag/prec,iso,debug
c      common/wf prof/htlist(101),lnlist(101,5),hlist(25),cflist(25,5),
$      nrlht,lht,nrmht,mht,charge(5),mratio(5),nrspec
c      common/itrat/dtheta(2),lub(2),maxitr,itr
c      common/m mtX/ml1,m21,m31,m12,m22,m32,m13,m23,m33
c      dimension en(5),nu(5),label(2),plotx(101,16),ploty(101),
$      ex(101), ey(101), ez(101), hx(101), hy(101), hz(101),
$      sex(101),sey(101),sez(101),shx(101),shy(101),shz(101),
$      sm11(101),sm21(101),sm31(101),
$      sm12(101),sm22(101),sm32(101),
$      sm13(101),sm23(101),sm33(101)
c      complex*16 mi/(0.d0,-1.d0)/,theta,thetap,st,stp,exc,templ,temp2,
$      ex,ey,ez,hx,hy,hz,sex,sey,sez,shx,shy,shz,
$      sm11,sm21,sm31,sm12,sm22,sm32,sm13,sm23,sm33,
$      ml1,m21,m31,m12,m22,m32,m13,m23,m33,e1,e2,e3,h1,h2,h3
c      real*8 magfld,nu,mratio,lnlist,lwstht,lub
c      real*4 plotx,ploty,capy,frq,rho,azm,cdp,sig,tpht,lwht,attn
c      logical first/.true./,savplt
c      integer debug,hdate(3),htime(2)
c      character*8 ctime,branch,bcd(10),label/'          ','total  '/
c      character*9 cdate
c
c      equivalence (cdate,hdate),(ctime,htime)
c      namelist/datum/h,alpha,prec,debug,nelect,
$      coefnu,expnu,charge,mratio,nprof,beta,hprime,sclhts,
$      topht,lwstht,itr,maxitr,dtheta,lub,
$      nprint,savplt
c
c      data dtr/1.745329252d-2/,npflag,nuflag/2*0/,
$      nprof/0/,beta,hprime/2*0.d0/,sclhts/5.d0/,
$      nprint/1/,savplt/.true./
c
c      data freq/0.d0/,azim/0.d0/,codip/0.d0/,magfld/0.d0/,
$      coefnu/1.816d11,4*4.54d9/,expnu/5*-.15d0/,
$      topht/100./,lwstht/0.d0/,nrht/101/,h/0.d0/,alpha/3.14d-4/,
$      sigma/5.d0/,epsr/81.d0/,nelect/0/,
$      prec/3.d-5/,debug/0/,
$      charge/-1.d0,1.d0,-1.d0,1.d0,-1.d0/,mratio/1.d0,4*5.8d4/,
$      itr/0/,maxitr/10/,dtheta/5.d-2,1.d-2/,lub/5.d-2,5.d-3/
c

```

```

      call date(cdate)
      call time(ctime)
      print 1000,cdate,ctime
c
10  read(5,180,end=999) bcd
      print 181,bcd
      if(bcd(1) .eq. 'quit'      ) .or.
$    bcd(1) .eq. 'QUIT'      ) go to 999
      if(bcd(1) .eq. 'name'     ) .or.
$    bcd(1) .eq. 'NAME'      ) go to 20
      if(bcd(1) .eq. 'profile'  ) .or.
$    bcd(1) .eq. 'PROFILE'   ) go to 30
      if(bcd(1) .eq. 'colfreq'  ) .or.
$    bcd(1) .eq. 'COLFREQ'   ) go to 50
      if(bcd(1) .eq. 'data'     ) .or.
$    bcd(1) .eq. 'DATA'      ) go to 60
      if(bcd(1) .eq. 'ipsq xmt' ) .or.
$    bcd(1) .eq. 'IPSQ XMT'  ) go to 60
      if(bcd(1) .eq. 'sw xmt'  ) .or.
$    bcd(1) .eq. 'SW XMT'    ) go to 60
      go to 910
c
c      namelist input
20  read(5,datum)
      print datum
      capk=1-.5*alpha*h
      if(nprof .eq. 0) go to 25
      if(beta*hprime .eq. 0.d0) go to 913
      npflag=1
      nn=hprime+sclhts/beta+.5d0
      nrspec=1
      c0=dlog(7.85535d-05*coefnu(1))-beta*hprime
      c1=beta*expnu(1)
      nrlht=2
      htlist(1)=topht
      htlist(2)=0.d0
      do 24 l=1,2
24  lnlist(l,1)=c0+c1*htlist(l)
c
c      number of points set to 101
25  delht=(topht-lwstht)/100.d0
26  if(nuflag .eq. 1) go to 10
      nrmht=2
      hlist(1)=topht
      hlist(2)=0.d0
      do 27 k=1,nrspec
      cflist(2,k)=dlog(coefnu(k))
27  cflist(1,k)=cflist(2,k)+topht*expnu(k)
      go to 10
c
c      profile input
30  decode(9,170,bcd) branch,nn
      npflag=1
      nrspec=nn
      if(nrspec .eq. 0) nrspec=1
      read(5,180) bcd
      print 181,bcd
      l=0

```

```

31  read(5,140) ht,en
    if(ht .lt. 0.d0) go to 39
    if(l .eq. 101) go to 912
    if(l .eq. 0) go to 32
    if(ht .ge. htlist(l)) go to 911
32  l=l+1
    if(nrspec .eq. 3) en(3)=en(2)-en(1)
    print 141,ht,(en(k),k=1,nrspec)
    htlist(l)=ht
    do 33 k=1,nrspec
33  lnlist(l,k)=dlog(dmax1(en(k),1.d-20))
    go to 31
39  nrlht=l
    go to 10

c
c  collision frequency profiles
50  nuflag = 1
    l=0
51  read(5,140) ht,nu
    if(ht .lt. 0.d0) go to 59
    if(l .eq. 25) go to 912
    if(l .eq. 0) go to 52
    if(ht .ge. hlist(l)) go to 911
52  l=l+1
    print 141,ht,(nu(k),k=1,nrspec)
    hlist(l)=ht
    do 53 k=1,nrspec
53  cflist(l,k)=dlog(dmax1(nu(k),1.d-20))
    go to 51
59  nrmht=l
    go to 10

c
c  wavefields calculations
60  if(npflag .eq. 0) go to 915
    read(5,180) bcd
61  read(5,1010) r,f,a,c,b,s,e
    if(r .eq. 40.d0) go to 300
    if(s .eq. 0.d0) go to 61
    rho=r
    freq=f
    azim=a
    codip=c
    magfld=b
    b=b*1.d4
    dcl= dsin(codip*dtr)*dcos(azim*dtr)
    dcm= dsin(codip*dtr)*dsin(azim*dtr)
    dcn=-dcos(codip*dtr)
    sigma=s
    epsr=e
    omega=6.28318530718d03*freq
    coefx=3.182357d09/omega**2
    capy=1.758796d11*magfld/omega
    wn=20.95845d0*freq
    const=-freq
    aconst=-8.686d0*wn
    econst=-.035d0*wn
    nsum=1
    nm=0

```

```

        if(nprint .le. 1) go to 62
        print 160
        print 181,bcd
        print 1011,r,f,a,c,b,s,e,h
62      read(5,1023) indx1,tr1,t1l,tmp1,tmpil,tmp2,tmpi2
        if(tr1 .eq. 0.d0) go to 91
        nm=nm+1
        read(5,1023) indx2,tr2,t2l,tmp3,tmpi3,tmp4,tmpi4
        if(tr1 .ne. tr2 .or. t1l .ne. t2l) go to 916
        thetap=dcmplx(tr1,t1l)
        if(tmp1 .eq. 0.d0 .and. tmpil .eq. 0.d0) tmp1=1.d-20
        temp1=dcmplx(tmp1,tmpil)
        if(indx1 .eq. 1) go to 63
        if(tmp3 .eq. 0.d0 .and. tmpi3 .eq. 0.d0) tmp3=1.d-20
        temp1=dcmplx(tmp3,tmpi3)
c
63      stp=cdsin(thetap*dtr)
        sr=stp
        si=stp*mi
        atten=aconst*si
        voverc=1.d0/sr
        exc=(-2.124292957d0,0.d0)*temp1*stp*stp
        excr=dcmplx(0.d0,econst)*exc
        exci=dcmplx(0.d0,econst)*exc*mi
        wm=10.d0*dlog10(excr**2+exci**2)
        wa=atan2(exci,excr)
        if(h .eq. 0.d0) go to 64
        st=stp*capk
        theta=mi*cdlog(cdsqrt(1.d0-st*st)-mi*st)/dtr
        go to 65
64      theta=thetap
65      if(nprint .le. 1) go to 66
        print 160
        print 181,bcd
        print 1011,r,f,a,c,b,s,e,h
66      print 1040,nm,theta,atten,voverc,wm,wa,thetap
c
        call wavfld(ex,ey,ez,hx,hy,hz)
c
74      lht=0
        ht=topht
        temp2=exc
        if(nprint .gt. 1) go to 75
        if(nprint .gt. 0 .and. nsum .eq. 1) go to 80
75      print 900,label(nsum)
80      do 90 k=1,nrht
        j=nrht+1-k
        if(nsum .eq. 1) go to 81
        e1=sex(j)
        e2=sey(j)
        e3=sez(j)
        h1=shx(j)
        h2=shy(j)
        h3=shz(j)
        go to 82
81      e1=ex(j)
        e2=ey(j)
        e3=ez(j)

```

```

      h1=hx(j)
      h2=hy(j)
      h3=hz(j)
      if(nm .eq. 1) go to 83
82    m11=sm11(j)
      m21=sm21(j)
      m31=sm31(j)
      m12=sm12(j)
      m22=sm22(j)
      m32=sm32(j)
      m13=sm13(j)
      m23=sm23(j)
      m33=sm33(j)
      if(nsum .eq. 2) go to 84
      sex(j)=sex(j)+ex(j)*temp2
      sey(j)=sey(j)+ey(j)*temp2
      sez(j)=sez(j)+ez(j)*temp2
      shx(j)=shx(j)+hx(j)*temp2
      shy(j)=shy(j)+hy(j)*temp2
      shz(j)=shz(j)+hz(j)*temp2
      go to 84
83    call tmtrx(ht)
      sex(j)=ex(j)*temp2
      sey(j)=ey(j)*temp2
      sez(j)=ez(j)*temp2
      shx(j)=hx(j)*temp2
      shy(j)=hy(j)*temp2
      shz(j)=hz(j)*temp2
      sm11(j)=m11
      sm21(j)=m21
      sm31(j)=m31
      sm12(j)=m12
      sm22(j)=m22
      sm32(j)=m32
      sm13(j)=m13
      sm23(j)=m23
      sm33(j)=m33
c...  q and s are relative
84    q=dcmplx(0.d0,const)*(dconjg(m11*e1+m12*e2+m13*e3)*e1
      +dconjg(m21*e1+m22*e2+m23*e3)*e2
      +dconjg(m31*e1+m32*e2+m33*e3)*e3)
      suma=suma+q*delht
      sx=e2*dconjg(h3)-dconjg(h2)*e3
      sy=e3*dconjg(h1)-dconjg(h3)*e1
      sz=e1*dconjg(h2)-dconjg(h1)*e2
      exm=cdabs(e1)
      eym=cdabs(e2)
      ezm=cdabs(e3)
      hxm=cdabs(h1)
      hym=cdabs(h2)
      hzm=cdabs(h3)
      exa=cdang(e1)
      eya=cdang(e2)
      eza=cdang(e3)
      hxa=cdang(h1)
      hya=cdang(h2)
      hza=cdang(h3)
      if(nsum .eq. 1) go to 87

```

```

      ploty(k)=ht
c      joule heating term
      plotx(k,1)=dlog10(dmax1(1.d-10,dabs(q )))
c      attenuation
      plotx(k,2)=suma
c      poynting vector magnitudes
      plotx(k,3)=dlog10(dmax1(1.d-10,dabs(sx)))
      plotx(k,4)=dlog10(dmax1(1.d-10,dabs(sy)))
      plotx(k,5)=dlog10(dmax1(1.d-10,dabs(sz)))
      angl=datan2(sz,sx)/dtr
      ang2=datan2(sz,sy)/dtr
      ang3=dacos((dcl*sx+dcn*sy+dcn*sz)/sqrt(sx**2+sy**2+sz**2))/dtr
c      angles between poynting vectors
      plotx(k,6)=angl
      plotx(k,7)=ang2
      plotx(k,8)=ang3
      call wfdens(ht,en,nu)
      capx=coefx*en(1)
      capz=nu(1)/omega
c      ionospheric parameters
      plotx(k, 9)=dlog10(dmax1(1.d-10,dabs(capx)))
      plotx(k,10)=dlog10(dmax1(1.d-10,dabs(capz)))
c      magnitude of electric field vectors
      plotx(k,11)=dlog10(dmax1(1.d-10,exm))
      plotx(k,12)=dlog10(dmax1(1.d-10,eym))
      plotx(k,13)=dlog10(dmax1(1.d-10,ezm))
c      magnitude of magnetic field vectors
      plotx(k,14)=dlog10(dmax1(1.d-10,hxm))
      plotx(k,15)=dlog10(dmax1(1.d-10,hym))
      plotx(k,16)=dlog10(dmax1(1.d-10,hzm))
      if(nprint .gt. 0) go to 88
      go to 89
87      if(nprint .le. 1) go to 89
88      print 901,ht,exm,exa,eym,eya,ezm,eza,q,sx,sy,sz,angl,ang2,
$          hxm,hxa,hym,hya,hzm,hza
89      ht=ht-delht
      if(ht .lt. htlist(lht+1)) lht=lht+1
      if(ht .lt. hlist(mht+1)) mht=mht+1
90      continue
c
      if(nsum .ne. 2) go to 62
      go to 92
91      nsum=2
      suma=0.
      if(nprint .eq. 0) go to 74
      print 160
      print 181,bcd
      print 1011,r,f,a,c,b,s,e,h
      print 1012,capy,nm
      go to 74
c
92      if(savplt) then
c
          if(first) then
              first=.false.
              open(unit=10,type='new',form='unformatted')
          end if
          rho=r

```

```

        frq=freq
        azm=azim
        cdp=codip
        sig=sigma
        tpht=topht
        lwht=lwstht
        attn=suma
c      save plot data on a file for later
        write(10) bcd,hdate,htime,frq,rho,azm,cdp,sig,capy,
$          nrht,tpht,lwht,attn,plotx,ploty
c
        end if
        go to 61
c
300    if(nprint .gt. 0) print 160
        go to 10
c
910    print 9910
        go to 999
911    print 9911
        go to 999
912    print 9912
        go to 999
913    print 9913
        go to 999
914    print 9914
        go to 999
915    print 9915
        go to 999
916    print 9916
999    stop
140    format(f7.2,4x,5(1x,e9.2))
141    format(f8.2,4x,1p5e9.2)
160    format(1h1)
161    format(1h )
170    format(1a8,1i1)
171    format(1x,1a8)
180    format(10a8)
181    format(1x,10a8)
900    format('0',a5,' Field strengths computed by WAVFLD'/
$        '0  ht',6x,'mag -- x -- ang  mag -- y -- ang',
$        3x,'mag -- z -- ang  q',10x,'sx',9x,'sy',9x,'sz',
$        12x,'alpha',6x,'beta')
901    format(1x,f5.1,' e',3(1p11.3,0pf7.3),1p4e11.3,0p2f11.3/
$        8x,' h',3(1p11.3,0pf7.3))
1000   format(' Additional plot identification: ',a9,2x,a8/)
1010   format(1x,f7.3,2x,f8.4,2x,f8.3,2x,f8.3,2x,e10.3,2x,e10.3,2x,f5.1)
1011   format('0r',f7.3,' f',f8.4,' a',f8.3,' c',f8.3,' m',f6.3,
$        ' s',1p10.3,' e',0pf5.1,' h',f5.1)
1012   format('+',72x,' mag(y)',1p10.3,' modes',i2)
1023   format(i1,2f9.0,1x,4e15.0)
1024   format('0input for nm = ',i2,': ',i1,0p2f10.5,1p2e16.8/
$        21x,' ',i1,0p2f10.5,1p2e16.8)
1040   format('0mode    real    imag    atten    v/c',9x,
$        'wait''s exc    real''    imag''')/
$        1x,i3,f11.5,f10.5,f9.3,f9.5,f10.3,f7.3,f10.5,f10.5)
9910   format('/' Error in control string')
9911   format('/' Heights in profile out of order')

```

```
9912 format(/' Too many heights in profile')
9913 format(/' BETA or HPRIME are 0.0')
9914 format(/' Number of output heights is greater than 101')
9915 format(/' No profile specified')
9916 format(/' Input data out of order')
end
```

```

      subroutine wavfld(ex,ey,ez,hx,hy,hz)
c
c  wavfld calls for the downward integration, and then performs the
c  back substitution of normalizing values (saved as data by wfstor).
c  field strengths are computed at heights from topht to lwstht at
c  delht increments and are returned in ex, ey, ez, hx, hy, hz.
c
      implicit real*8 (a-h,o-z)
      common/wf inpt/theta,freq,azimuth,codip,magfld,ceffnu(5),expnu(5),
$      topht,lwstht,delht,h,alpha,sigma,epsilon,nglect
      common/wf flag/precn,iso,idbg
      common/wf save/p(4,2),m31,m32,m33,ortho,anorm,bnorm,ht,levl
      common/cs/c,s,ci,si
      real*8 magfld,lwstht
      complex*16 theta,ex(1),ey(1),ez(1),hx(1),hy(1),hz(1),
$      p,m31,m32,m33,ortho,c,s,ci,si,b(2),w(4),osum
c
      jht=topht/delht+1.01d0
      test=(jht-1)*delht-topht
      if(dabs(test) .gt. 1.d-4) go to 800
      mht=lwstht/delht+1.01d0
      test=(mht-1)*delht-lwstht
      if(dabs(test) .gt. 1.d-4) go to 800
      jht=min0(jht-mht+1,101)
      mht=1
c
c  iteration to satisfy modal equation
      call itrata
c
c  combine solutions at ground so that they satisfy boundary condition.
      call wf bndy(b)
c
c  perform back substitution of normalizing values.
20 o sum=0.0
      proda=1.0
      prodb=1.0
      iht=mht
      call wf load
      go to 25
c
21 o sum=o sum*anorm/bnorm+ortho
      proda=proda*anorm
      if(proda .lt. 1.0d-30) proda=0.0
      prodb=prodb*bnorm
      call wf load
      do 23 j=1,4
        p(j,2)=(p(j,2)-o sum*p(j,1))*prodb
23 p(j,1)=p(j,1)*proda
c
c  compute field strengths at profile heights.
25 do 26 j=1,4
26 w(j)=p(j,1)*b(1)+p(j,2)*b(2)
      ex(iht)=w(1)
      ey(iht)=-w(2)
      ez(iht)=-(s*w(4)+m31*w(1)-m32*w(2))/(1.0+m33)
      hx(iht)=w(3)
      hy(iht)=w(4)
      hz(iht)=-s*w(2)

```

```

c      iht=iht+1
      if(iht .le. jht) go to 21
      if(levl .ne. 0) print 903,levl
      return
c
800 print 902
      stop
c
902 format('0error in wavfld'/
$      ' delht does not divide topht-lwstht evenly')
903 format('0levl not zero: levl=',i3)
      end

```

```

      subroutine wf intg(topht,lwstht,delht,iflag)
c
c   wf intg performs the integration of the p matrix down through the
c   ionosphere, using the techniques given by pitteway.
c   accuracy is maintained by adjusting the stepsize so that the
c   p matrix is computed with sufficient accuracy.
c
c   iflag=0  integ for theta only
c   iflag=1  integ for theta and theta-dtheta
c
      implicit real*8 (a-h,o-z)
      common/wf flag/precn,iso,idbg
      common/p  mtx/p(16),pi(16)
      commoncn/wf save/p save(16),m31 sav,m32 sav,m33 sav,
$           ortho,anorm,bnorm,ht,levl
      common/m  mtx/m(3,3)
      common/wf prof/enht(101),enlog(101,5),collht(25),collfr(25,5),
$           nrlht,lht,nrmht,mht,charge(5),ratiom(5),nrspec
      integer svflag
      real*8 lwstht
      complex*16 m31 sav,m32 sav,m33 sav,ortho,m
      dimension prevp(16),tempp(16),dpdh(16),pv dpdh(16),dpidh(16)
c
c   minimum step-size allowed
c   data epsht/5.d-4/,dhmin/1.d-3/
c
      call init t
      call t mtrx(top ht)
      call wf init(p)
      call p deriv(p,dpdh)
      if(iflag .eq. 0) go to 11
      call ti mtrx
      call wf init(pi)
      call p deriv(pi,dpidh)
11  continue
c
      isteps=0
      kmax=0
      lev!=0
      ht=topht
      call xfer(p,p save,16)
      m31 sav=m(3,1)
      m32 sav=m(3,2)
      m33 sav=m(3,3)
      call wf stor
      wfht=topht-delht
      delh2=0.125d0*delht
      svflag=0
c
c   determine next stepsize to use.
10  if(svflag .eq. 1) delh2=savdh2
      svflag=0
      nodbl=0
      ht0=ht
      call xfer(p,prevp,16)
      call xfer(dpdh,pv dpdh,16)
      htlim=wfht
      if(enht(lht+1) .gt. htlim+epsht) htlim=enht(lht+1)

```

```

        if(collht(mht+1) .gt. htlim+epsht) htlim=collht(mht+1)
        if(ht0-delh2.ge.htlim+epsht) go to 50
        savdh2=delh2
        svflag=1
        delh2=ht0-htlim
c
c   perform next integration step.
50 call wf step(p,dpdh,ht,delh2,0)
   call xfer(p,tempp,16)
   m31 sav=m(3,1)
   m32 sav=m(3,2)
   m33 sav=m(3,3)
   ht=ht0
   call xfer(prevp,p,16)
   call xfer(pv dpdh,dpdh,16)
   delh=0.5*delh2
   call wf step(p,dpdh,ht,delh,1)
   call p deriv(p,dpdh)
   call wf step(p,dpdh,ht,delh,2)
c   check accuracy of result.
   pmax=0.0
   do 85 j=1,16
     pabs=dabs(p(j)-tempp(j))
     if(pmax .lt. pabs) pmax=pabs
85 continue
c   adjust stepsize if necessary.
   if(pmax .lt. precsn) go to 100
c
   if(delh .gt. dhmin) go to 95
   if(kmax .eq. 0) print 900, ht
   kmax=1
   go to 100
95 continue
   delh2=0.5*delh2
   nodbl=1
   if(pmax .lt. 10.0*precsn) go to 99
   delh2=0.25*delh2
   nodbl=0
99 continue
   ht=ht0
   call xfer(prevp,p,16)
   call xfer(pv dpdh,dpdh,16)
   svflag=0
   go to 50
c
100 call wf scal(p,0)
   call xfer(p,p save,16)
   if(ht .lt. wfht+epsht) call wf stor
   call p deriv(p,dpdh)
   if(iflag .eq. 0) go to 72
   ht=ht0
   call wf step(pi,dpidh,ht,delh,3)
   call p deriv(pi,dpidh)
   call wf step(pi,dpidh,ht,delh,4)
   call wf scal(pi,1)
   call p deriv(pi,dpidh)
72 continue
c

```

```

        isteps=isteps+1
        if(idbg .eq. 0) go to 73
        idiv=isteps/50
        if(isteps .eq. 50*idiv) print 902,isteps,ht
73      continue
        if(no dbl .eq. 0 .and. pmax .lt. 0.1*precsn) delh2=2.0*delh2
c
c      check integration and profile heights.
        if(ht .lt. lwtstht+epsht) go to 80
        if(ht .lt. wfht+epsht) wfht=wfht-delht
        if(ht .lt. enht(lht+1)+epsht) lht=lht+1
        if(ht .lt. collht(mht+1)+epsht) mht=mht+1
        go to 10
c
80      print 901,isteps
        return
c
900      format('0minimum stepsize used at ht=',1pe14.5)
901      format('0',i4,' integration steps used in wavfld')
902      format('0',i4,' integration steps, ht=',f9.4)
        end

```

```

      subroutine wf scal(pp,iflag)
c
c   wfscal scales and orthogonalizes the solution vectors p.
c   this scaling must later be removed to yield correct (unscaled)
c   solutions.
c
      implicit real*8 (a-h,o-z)
      common/wf save/p save(16),m31 sav,m32 sav,m33 sav,
$      o sum,aprod,bprod,ht,levl
      common/save/p etc(27,101)
      complex*16 p(4,2),m31 sav,m32 sav,m33 sav,o sum,ortho
      dimension pr(8,2),pp(16)
      equivalence (p,pr)
c
      call xfer(pp,pr,16)
      anorm=0.0
      do 11 j=1,8
11  anorm=anorm+pr(j,1)**2
      ortho=0.0
      do 12 j=1,4
12  ortho=ortho+dconjg(p(j,1))*p(j,2)
      ortho=ortho/anorm
      do 13 j=1,4
13  p(j,2)=p(j,2)-ortho*p(j,1)
      bnorm=0.0
      do 14 j=1,8
14  bnorm=bnorm+pr(j,2)**2
      anorm=1.0/dsqrt(anorm)
      bnorm=1.0/dsqrt(bnorm)
      do 15 j=1,8
      pr(j,1)=pr(j,1)*anorm
15  pr(j,2)=pr(j,2)*bnorm
      call xfer(pr,pp,16)
      if(iflag .ne. 0) return
      o sum=o sum+ortho*aprod/bprod
      aprod=aprod*anorm
      bprod=bprod*bnorm
      return
c
c                                     entry wf stor
      entry wf stor
      levl=levl+1
      call xfer(p save,p etc(1,levl),27)
      o sum=0.0
      a prod=1.0
      b prod=1.0
      return
c
c                                     entry wf load
      entry wf load
      call xfer(p etc(1,levl),p save,27)
      levl=levl-1
      return
      end

```

```

      subroutine wf step(p,dpdh,ht,delh,iflag)
c
c   wf step increments the solution of p from ht to ht-delh,
c   using runge-kutta integration
c
c   iflag=0  one large step, theta
c   iflag=1  first small step, theta
c   iflag=2  second small step, theta
c   iflag=3  first small step, theta-dtheta
c   iflag=4  second small step, theta-dtheta
c
      implicit real*8 (a-h,o-z)
      common/wf con/omega,wave nr
      common/wf flag/precn,iso,idbg
      common/t mtz/t(18)
      common/tm mtz/tm(18)
      dimension p(16),dpdh(16),p0(16),
$             hdelp0(16),delp1(16),delp2(16)
      dimension t save1(18),t save2(18),
$             tm sav1(18),tm sav2(18),tm sav3(18),tm sav4(18)
c
      ht0=ht
      delh k=delh*wave nr
      hdelh k=delh k*0.5
      do 11 j=1,16
      p0(j)=p(j)
      hdelp0(j)=-dpdh(j)*hdelh k
11  p(j)=p0(j)+hdelp0(j)
c
      ht=ht0-0.5*delh
      if(iflag .le. 2) call t mtrx(ht)
      if(iflag .eq. 0) call xfer(t,t save1,18)
      if(iflag .eq. 0) call xfer(tm,tm sav2,18)
      if(iflag .eq. 1) call xfer(tm,tm sav1,18)
      if(iflag .eq. 2) call xfer(tm,tm sav3,18)
      if(iflag .eq. 3) call xfer(tm sav1,tm,18)
      if(iflag .eq. 4) call xfer(tm sav3,tm,18)
      if(iflag .ge. 3) call ti mtrx
c
      call p deriv(p,dpdh)
      do 12 j=1,16
      delp1(j)=-dpdh(j)*delh k
12  p(j)=p0(j)+0.5*delp1(j)
c
      call p deriv(p,dpdh)
      do 13 j=1,16
      delp2(j)=-dpdh(j)*delh k
13  p(j)=p0(j)+delp2(j)
c
      ht=ht0-delh
      if(iflag .eq. 0) call t mtrx(ht)
      if(iflag .eq. 0) call xfer(t,t save2,18)
      if(iflag .eq. 1) call xfer(t save1,t,18)
      if(iflag .eq. 2) call xfer(t save2,t,18)
      if(iflag .eq. 0) call xfer(tm,tm sav4,18)
      if(iflag .eq. 3) call xfer(tm sav2,tm,18)
      if(iflag .eq. 4) call xfer(tm sav4,tm,18)
      if(iflag .ge. 3) call ti mtrx

```

c

```
      call p deriv(p,dpdh)
      third=1.0d0/3.0d0
      do 14 j=1,16
      delp4=(hdel p0(j)+del p1(j)+del p2(j)-dpdh(j)*hdel h k)*third
14  p(j)=p0(j)+delp4
      return
      end
```

```

      subroutine xfer(a,b,n)
c
c  transfer array a into array b.
c
      real*8 a,b
      dimension a(1),b(1)
c
      do 11 j=1,n
11  b(j)=a(j)
      return
      end

```

```

      subroutine wf dens (ht, en, coll)
c
c   wf dens computes the ion density and collision frequency for each
c   specie by logarithmic interpolation of the corresponding profiles.
c   profile values are interpolated between entries mht and mht+1
c   (lht and lht+1).
c
      implicit real*8 (a-h,o-z)
      common/wf prof/enht(101),enlog(101,5),collht(25),collfr(25,5),
$      nrlht,lht,nrmht,mht,charge(5),ratiom(5),nrspec
      dimension en(5), coll(5), dele(5), delc(5)
c
      if(lht .eq. 0) then
        lsave=0
10      lht=lht+1
        if(ht .lt. enht(lht)) go to 10
        mht=0
        msave=0
20      mht=mht+1
        if(ht .lt. collht(mht)) go to 20
      end if
      if(lht .ne. lsave) then
        if(lht .ge. nrlht) lht=nrlht-1
        delh=enht(lht+1)-enht(lht)
        do 150 k=1,nrspec
150      dele(k)=(enlog(lht+1,k)-enlog(lht,k))/delh
        lsave=lht
      end if
      if(mht .ne. msave) then
        if(mht .ge. nrmht) mht=nrmht-1
        delh=collht(mht+1)-collht(mht)
        do 250 k=1,nrspec
250      delc(k)=(collfr(mht+1,k)-collfr(mht,k))/delh
        msave=mht
      end if
      dh=ht-enht(lht)
      dc=ht-collht(mht)
      do 500 k=1,nrspec
        en(k)=dexp(enlog(lht,k)+dh*dele(k))
500      coll(k)=dexp(collfr(mht,k)+dc*delc(k))
      return
      end

```

```

      subroutine wf init(p)
c
c   wf init computes the initial p matrix, i.e., the initial conditions
c   for the integration  $dp/dz = -ik \cdot t \cdot p$ .
c
      implicit real*8 (a-h,o-z)
      common/t mtx/t11,t31,t41,t12,t32,t42,t14,t34,t44
      common/wf flag/precn,iso,idbg
      complex*16 b3,b2,b1,b0,i,q temp,det,sqroot,q(4),zp,
$          t11,t31,t41,t12,t32,t42,t14,t34,t44
      dimension p(8,2)
      data i/(0.d0,1.d0)/
c
      if(iso .ne. 0) go to 50
c
      b3=-(t11+t44)
      b2=t11*t44-t14*t41-t32
      b1=(-t32*(t11+t44)+t12*t31+t34*t42)
      b0=-t11*(t32*t44-t34*t42)
$      +t12*(t31*t44-t34*t41)
$      -t14*(t31*t42-t32*t41)
      call quartc(b3,b2,b1,b0,q)
c
      qi min=-i*q(1)
      j1=1
      do 22 j=2,4
      q imag=-i*q(j)
      if(q imag .gt. qi min) go to 22
      qi min=q imag
      j1=j
22 continue
c
      qr max=q(1)
      j2=1
      do 23 j=2,4
      q real=q(j)
      if(q real .lt. qr max) go to 23
      qr max=q real
      j2=j
23 continue
c
      if(j1 .eq. j2) go to 80
c
      q temp=q(j2)
      q(1)=q(j1)
      q(2)=q temp
c
      do 31 j=1,2
      det=(t11-q(j))*(t44-q(j))-t14*t41
      zp=(t12*q(j)-(t12*t44-t14*t42))/det
      p(1,j)=dreal(zp)
      p(2,j)=dimag(zp)
      p(3,j)=1.0
      p(4,j)=0.0
      zp=q(j)
      p(5,j)=dreal(zp)
      p(6,j)=dimag(zp)
      zp=(t42*q(j)+(t12*t41-t11*t42))/det

```

```

      p(7,j)=dreal(zp)
31  p(8,j)=dimag(zp)
c
40  if(idbg .lt. 2) return
    print 902,q
    print 901,p
    return
c
50  b1=(t11+t44)*0.5
    b0=t11*t44-t14*t41
    sqroot=cdsqrt(b1**2-b0)
    q(1)=b1+sqroot
    q(4)=b1-sqroot
    sqroot=cdsqrt(t32)
    q(2)=+sqroot
    q(3)=-sqroot
c
    q1 test=q(1)+i*q(1)
    q4 test=q(4)+i*q(4)
    if(q4 test .gt. q1 test) q(1)=q(4)
    q2 test=q(2)+i*q(2)
    q3 test=q(3)+i*q(3)
    if(q3 test .gt. q2 test) q(2)=q(3)
c
    zp=t14
    p(1,1)=dreal(zp)
    p(2,1)=dimag(zp)
    p(3,1)=0.0
    p(4,1)=0.0
    p(5,1)=0.0
    p(6,1)=0.0
    zp=-(t11-q(1))
    p(7,1)=dreal(zp)
    p(8,j)=dimag(zp)
c
    p(1,2)=0.0
    p(2,2)=0.0
    p(3,2)=0.0
    p(4,2)=0.0
    zp=q(2)
    p(5,2)=dreal(zp)
    p(6,2)=dimag(zp)
    p(7,2)=0.0
    p(8,2)=0.0
    go to 40
c
80  print 900,q
    stop
900 format('0error in wf init, q values do not sort'/
$      , q=',4(1pe15.5,1pe13.5))
901 format('0p values='/'
$      4(1pe15.5,1pe13.5)/4(1pe15.5,1pe13.5))
902 format('0initial values from wf init at ht=topht'/
$      , q=',2(1pe15.5,1pe13.5))
    end

```

```

      subroutine quartc (fourb3, sixb2, fourb1, b0, q)
c
c   quartc finds the roots of a quartic polynomial from the closed form.
c
      implicit real*8 (a-h,o-z)
      complex*16 b3,b2,b1,b0,q,fourb3,sixb2,fourb1,b3sq,h,i,g,hprime,
$          gprime,sqroot,pplus,p,logp,cbert0,cbert1,cbert2,omegal,
$          omega2,rootp,rootq,rootr
      real*8 mgplus,mgmnus
      dimension q(4),pri(2)
      equivalence (p,pri)
c
      data omegal/(-.5d0, .8660254038d0)/,omega2/(-.5d0,-.8660254038d0)/
      data precsn/1.d-10/
c
      b3=fourb3*0.25
      b2=sixb2/6.0
      b1=fourb1*0.25
      b3sq=b3**2
      h=b2-b3sq
      i=b0-4.0*b3*b1+3.0*b2**2
      g=b1+b3*(-3.0*b2+2.0*b3sq)
      hprime=-i/12.0
      gprime=-g**2/4.0-h*(h**2+3.0*hprime)
      sqroot=cdsqrt(gprime**2+4.0*hprime**3)
      p=(-gprime+sqroot)*0.5
      mgplus=dabs(pri(1))+dabs(pri(2))
      pplus=p
      p=(-gprime-sqroot)*0.5
      mgmnus=dabs(pri(1))+dabs(pri(2))
      if(mgplus .gt. mgmnus) p=pplus
      logp=cdlog(p)
      cbert0=cdexp(logp/3.0)
      cbert1=omegal*cbert0
      cbert2=omega2*cbert0
      rootp=cdsqrt(cbert0-hprime/cbert0-h)
      rootq=cdsqrt(cbert1-hprime/cbert1-h)
      rootr=cdsqrt(cbert2-hprime/cbert2-h)
      if(cdabs(g) .le. 1.0d-20) go to 5
      sign=-rootp*rootq*rootr*2.0/g
      if(sign .lt. 0.0) rootr=-rootr
c
c   5
      q(1)=+rootp+rootq+rootr-b3
      q(2)=+rootp-rootq-rootr-b3
      q(3)=-rootp+rootq-rootr-b3
      q(4)=-rootp-rootq+rootr-b3
c
      do 20 n=1,4
      iter=0
10      rootp=q(n)**4+fourb3*q(n)**3+sixb2*q(n)**2+fourb1*q(n)+b0
      rootq=4.0*q(n)**3+3.0*fourb3*q(n)**2+2.0*sixb2*q(n)+fourb1
      rootr=rootp/rootq
      q(n)=q(n)-rootr
      if(cdabs(rootr) .lt. precsn) go to 20
      iter=iter+1
      if(iter .lt. 10) go to 10
      print 900, iter, q(n)
20      continue

```

```
c      return
900    format(i3,' iterations, q=',e15.5,e13.5,' fails to converge')
      end
```

```

subroutine rbars(c,s,rbar11,rbar22,ey,hy)
c
  implicit real*8 (a-h,o-z)
  common/wf inpt/theta,freq,azim,codip,magfld,coefnu(5),expnu(5),
$          topht,lwstht,delht,h,alpha,sigma,epsr,nelect
  common/wf con/omega,k
  common/ey grnd/eyg,hyg
  complex*16 theta,i,ngsq,c,s,ssq,sqroot,rtiort,ikc,
$          p0,h10,h20,h1prm0,h2prm0,caph10,caph20,
$          pd,h1d,h2d,h1prmd,h2prmd,caph1d,caph2d,
$          pz,h1z,h2z,h1prmz,h2prmz,
$          alst,a2nd,a3rd,a4th,a1,a2,a3,a4,
$          exd,exdsq,exz,exzsq,
$          rbar11,rbar22,z1,z2,
$          den12,den34,
$          eyg,hyg,
$          ex,ey,ez,hx,hy,hz
  real*8 k,kvraot,kvratt,n0sq,ndsq,nzsq,magfld,lwstht
  equivalence (pz,pd),(h1z,h1d),(h2z,h2d),(h1prmz,h1prmd),
$          (h2prmz,h2prmd),(exd,exz),(exdsq,exzsq)
c
  data i/(0.0d0,1.0d0)/
  data tstthm/10.d0/
  data epsln0/8.85434d-12/
c
  eyg=1.0
  hyg=1.0
  alt=lwstht
  d=lwstht
  ssq=s*s
  ngsq=dcmplx(epsr,-sigma/(omega*epsln0))
  sqroot=cdsqrt(ngsq-ssq)
  thtim=i*theta
  if(thtim.gt. tstthm) go to 10
c
  kvraot=dexp(dlog(k/alpha)/3.0)
  kvratt=kvraot**2
  avrkot=1.0/kvraot
  avrktt=avrkot**2*0.5
  n0sq=1.0-alpha*h
  rtiort=n0sq/ngsq*sqroot
  p0=kvratt*(n0sq-ssq)
  call mdhnl (p0,h10,h20,h1prm0,h2prm0,theta,'rb 1')
  caph10=h1prm0+avrktt*h10
  caph20=h2prm0+avrktt*h20
  alst=caph20-i*rtiort*kvraot*h20
  a2nd=caph10-i*rtiort*kvraot*h10
  a3rd=h2prm0-i*kvraot*sqroot*h20
  a4th=h1prm0-i*kvraot*sqroot*h10
  den12=h20*a2nd-h10*alst
  den34=h20*a4th-h10*a3rd
  if(d.eq. 0.0) go to 10
c
  ndsq=1.0-alpha*(h-d)
  pd=kvratt*(ndsq-ssq)
  call mdhnl (pd,h1d,h2d,h1prmd,h2prmd,theta,'rb 2')
  caph1d=h1prmd+avrktt*h1d
  caph2d=h2prmd+avrktt*h2d

```

```

c
a1=c*nds*(h2d*a2nd-h1d*alst)
a2=i*avrkot*(caph1d*alst-caph2d*a2nd)
a3=i*avrkot*(h2prmd*a4th-h1prmd*a3rd)
a4=c*(h2d*a4th-h1d*a3rd)
rbar11=(a1-a2)/(a1+a2)
rbar22=(a3+a4)/(a4-a3)
expon=dexp(0.5*alpha*alt)
hy=(h2z*a2nd-h1z*alst)*expon/den12*hyg
ey=(h2z*a4th-h1z*a3rd)/den34*eyg
return

c
c flat earth
10 ikc=i*k*c
exd=cdexp(-ikc*d)
exdsq=exd**2
z1=(ngsq*c-sqroot)/(ngsq*c+sqroot)
z2=(c-sqroot)/(c+sqroot)
rbar11=z1*exdsq
rbar22=z2*exdsq
hy=(1.0+z1*exzsq)/(1.0+z1)/exz*hyg
ey=(1.0+z2*exzsq)/(1.0+z2)/exz*eyg
return

c entry wf htgn(ht,ex,ey,ez,hx,hy,hz,c,s)
alt=ht
if(thtim.gt. tstthm) go to 50
nzsqu=1.0-alpha*(h-alt)
pz=kvratt*(nzsqu-ssq)
call mdhnl(pz,h1z,h2z,h1prmd,h2prmd,theta,'htgn')
expon=dexp(0.5*alpha*alt)
hy=(h2z*a2nd-h1z*alst)*expon/den12*hyg
ey=(h2z*a4th-h1z*a3rd)/den34*eyg
ex=i*avrkot*((h2prmd*a2nd-h1prmd*alst)/
$ den12*hyg*expon+avrkt*hy)/nzsqu
ez=-s/nzsqu*hy
hz=s*ey
hx=avrkot/i*(h2prmd*a4th-h1prmd*a3rd)/den34*eyg
return

c
c flat earth
50 exz=cdexp(-ikc*alt)
exzsq=exz**2
hy=(1.0+z1*exzsq)/(1.0+z1)/exz*hyg
ey=(1.0+z2*exzsq)/(1.0+z2)/exz*eyg
ex=-c*(1.0-z1*exzsq)/(1.0+z1)/exz*hyg
ez=-s*hy
hz=s*ey
hx=c*(1.0-z2*exzsq)/(1.0+z2)/exz*eyg
return
end

```

```

      subroutine r mtrx(p,cosn,r)
c
c   r mtrx computes reflection coefficient matrix from p matrix
c   and returns it in r.
c
      implicit real*8 (a-h,o-z)
      complex*16 zp(4,2),cosn,r(2,2),
$          gl2, gl3, gl4, g23, g24, g34,
$          d00, d11, d22, d12, d21
      dimension p(8,2)
c
      do 2 j=1,2
      zp(1,j)=dcmplx(p(1,j),p(2,j))
      zp(2,j)=dcmplx(p(3,j),p(4,j))
      zp(3,j)=dcmplx(p(5,j),p(6,j))
2      zp(4,j)=dcmplx(p(7,j),p(8,j))
c
      gl2=zp(1,1)*zp(2,2)-zp(1,2)*zp(2,1)
      gl3=zp(1,1)*zp(3,2)-zp(1,2)*zp(3,1)
      gl4=zp(1,1)*zp(4,2)-zp(1,2)*zp(4,1)
      g23=zp(2,1)*zp(3,2)-zp(2,2)*zp(3,1)
      g24=zp(2,1)*zp(4,2)-zp(2,2)*zp(4,1)
      g34=zp(3,1)*zp(4,2)-zp(3,2)*zp(4,1)
c
      d00=-gl3+cosn*( g34-gl2+cosn*g24)
      d11= gl3+cosn*( g34+gl2+cosn*g24)
      d22= gl3+cosn*(-g34-gl2+cosn*g24)
      d12=2.0*cosn*gl4
      d21=2.0*cosn*g23
c
      r(1,1)=d11/d00
      r(2,2)=d22/d00
      r(1,2)=d12/d00
      r(2,1)=d21/d00
      return
      end

```

```

      subroutine t mtrx(ht)
c
c   t mtrx computes m- the susceptibility tensor and
c       t- the coefficient matrix of  $dp/dz = -ik \cdot t \cdot p$ .
c   note that on call to entry init t, various ionospheric
c   constants are computed.
c
      implicit real*8 (a-h,o-z)
      common/wf flag/precn,iso,idbg
      common/m mtz/ml1,m21,m31,m12,m22,m32,m13,m23,m33
      common/t mtz/tl1,t31,t41,t12,t32,t42,t14,t34,t44
      common/tm mtz/tm11,tm31,tm41,tm12,tm32,tm42,tm14,tm34,tm44
      common/cs/c,s,ci,si
      common/wf inpt/theta,freq,azmuth,codip,magfld,ceffnu(5),expnu(5),
$      toph, lwstht, delht, h, alpha, sigma, epsilon, nglect
      common/itrat/dtheta,dlub(2),maxitr,itr
      common/wf con/omega,wave nr
      common/wf prof/enht(101),enlog(101,5),collht(25),collfr(25,5),
$      nrlht,lht,nrmht,mht,charge(5),ration(5),nrsec
      real*8 magfld,lwstht,lsqysq,msqysq,nsqysq,lmysq,lnysq,mnysq,
$      nu,ly,my,ny
      complex*16 m(3,3),
$      ml1,m21,m31,m12,m22,m32,m13,m23,m33,
$      tl1,t31,t41,t12,t32,t42,t14,t34,t44,
$      tm11,tm31,tm41,tm12,tm32,tm42,tm14,tm34,tm44,
$      c,s,ci,si,csq,ssq,csqi,ssqi,
$      theta,dtheta,
$      d,ml3d,m23d,
$      u,usq,dd,i,iud,ta,tb
      dimension y(5),ysq(5),ly(5),my(5),ny(5),coef en(5),en(5),nu(5),
$      lmysq(5),lnysq(5),mnysq(5),lsqysq(5),msqysq(5),nsqysq(5)
      equivalence (ml1,m)
c
      data pi/3.141592653d0/
      data twopi/6.28318530717959d0/
      data dtr/0.01745329252d0/
      data coeffx/3.182357d03/,coeffy/1.758796d11/
      data i/(0.d0,1.d0)/
      data veilt/2.997928d05/
c
c   calculate the matrix m.
      m(1,1)=0.0
      m(1,2)=0.0
      m(1,3)=0.0
      m(2,1)=0.0
      m(2,2)=0.0
      m(2,3)=0.0
      m(3,1)=0.0
      m(3,2)=0.0
      m(3,3)=0.0
c
      call wf dens (ht, en, nu)
      nflag=0
      do 20 k=1,nrsec
c   add in the contributions to the susceptibility tensor m for each
c   specie in the ionosphere.
      if(en(k) .lt. 1.0d-3) go to 20
      nflag=1

```

```

x=coef en(k)*en(k)
if(nglect .ne. 0) x=-x
z=nu(k)*ov omga
if(nglect .ne. 0) z=-z
u=1.0-i*z
usq=u*u
dd=-x/(u*(usq-ysq(k)) )
iud=(z+i)*dd
ta=usq*dd
m(1,1)=m(1,1)+ta
m(2,2)=m(2,2)+ta
m(3,3)=m(3,3)+ta
m(2,2)=m(2,2)-msqysq(k)*dd
ta=my(k)*iud
tb=lnysq(k)*dd
m(1,3)=m(1,3)+ta-tb
m(3,1)=m(3,1)-ta-tb
if(iso .ne. 0) go to 20
m(1,1)=m(1,1)-lsqysq(k)*dd
m(3,3)=m(3,3)-nsqysq(k)*dd
ta=ny(k)*iud
tb=lmysq(k)*dd
m(2,1)=m(2,1)+ta-tb
m(1,2)=m(1,2)-ta-tb
ta=ly(k)*iud
tb=mnysq(k)*dd
m(3,2)=m(3,2)+ta-tb
m(2,3)=m(2,3)-ta-tb
20 continue
c
crvtrm=alpha*(h-ht)
m(1,1)=m(1,1)-crvtrm
m(2,2)=m(2,2)-crvtrm
m(3,3)=m(3,3)-crvtrm
c
c calculate the matrix t.
d=1.0/(1.0+m33)
tm41=1.0+m11
tm32=m22
tm14=d
if(nflag .eq. 0) go to 40
m13d=m13*d
m23d=m23*d
tm41=tm41-m31*m13d
tm11=m31*d
tm44=m13d
if(iso .ne. 0) go to 40
tm32=tm32-m32*m23d
tm31=m31*m23d-m21
tm12=m32*d
tm42=m32*m13d-m12
tm34=m23d
c
40 t41=tm41
t32=csq+tm32
t14=1.0-ssq*tm14
if(nflag .eq. 0) go to 70
t11=-s*tm11

```

```

t44=-s*tm44
if(iso .ne. 0) return
t31=tm31
t12=s*tm12
t42=tm42
t34=s*tm34
return
c entry ti mtrx
t41=tm41
t32=csqi+tm32
t14=1.0-ssqi*tm14
t11=-si*tm11
t44=-si*tm44
if(iso .ne. 0) return
t31=tm31
t12=si*tm12
t42=tm42
t34=si*tm34
return
c entry init t
lht=0
iso=0
if(magfld .eq. 0.d0) go to 250
if(dabs(codip-90.d0) .ge. 0.15d0) go to 300
if(dabs(azmuth- 90.d0) .lt. 0.15d0) go to 250
if(dabs(azmuth-270.d0) .ge. 0.15d0) go to 300
250 iso=1
300 omega=twopi*freq*1000.d0
ov omga=1.0/omega
wavenr=omega/vellt
sindip=dsin(codip*dtr)
drcosl=sindip*dcos(azmuth*dtr)
drcosm=sindip*dsin(azmuth*dtr)
drcosn=-dcos(codip*dtr)
do 60 k=1,nrspec
coef en(k)=coeffx*1.0d6*charge(k)**2/(omega**2*ratiom(k))
y(k)=coeffy*charge(k)*magfld/(omega*ratiom(k))
ysq(k)=y(k)**2
ly(k)=drcosl*y(k)
my(k)=drcosm*y(k)
ny(k)=drcosn*y(k)
lsqysq(k)=drcosl**2*ysq(k)
msqysq(k)=drcosm**2*ysq(k)
nsqysq(k)=drcosn**2*ysq(k)
lmysq(k)=drcosl*drcosm*ysq(k)
lnysq(k)=drcosl*drcosn*ysq(k)
mnysq(k)=drcosm*drcosn*ysq(k)
60 continue
c=cdcos(theta*dtr)
s=cdsin(theta*dtr)
csq=c**2
ssq=s**2
ci=cdcos((theta-dtheta)*dtr)
si=cdsin((theta-dtheta)*dtr)
csqi=ci**2
ssqi=si**2

```

c
70 t11=0.0
t31=0.0
t12=0.0
t42=0.0
t34=0.0
t44=0.0
return
end

```

      subroutine wf bndy(b)
c
c   wf bndy computes the vector b, which determines how to combine
c   the solution vectors in order to satisfy the boundary conditions.
c   this routine is valid only for eigenangles of the modal equation
c   and is used to compute height gain functions.
c
      implicit real*8 (a-h,o-z)
      common/wf flag/precsn,iso,idbg
      common/p mtx/p(4,2),pi(16)
      common/ey grnd/eyg,hyg
      complex*16 p,b(2),r(2,2),f,rbar11,rbar22,nurmf,denmf,uma,dena,a,
$           ex1,ex2,ey1,ey2,hx1,hx2,hy1,hy2,ey,hy,eyg,hyg,c,s,fofr,
$           hysum,abpar1,abperp
      dimension pp(16)
      data dtr/0.01745329252d0/
c
      if(idbg .gt. 1) print 902, p
      ex1=p(1,1)
      ex2=p(1,2)
      ey1=-p(2,1)
      ey2=-p(2,2)
      hx1=p(3,1)
      hx2=p(3,2)
      hy1=p(4,1)
      hy2=p(4,2)
c
      if(iso .ne. 0) go to 500
c   compute b, non-isotropic case (from polarization ey/hy, see budden).
      nurmf=r(2,1)*(1.0+rbar22)*rbar11
      denmf=(1.0+rbar11)*(1.0-rbar22*r(2,2))
      f of r=nurmf/denmf
      uma=-(ey1-fofr*hy1)
      dena=ey2-fofr*hy2
      a=uma/dena
      hysum=hy1+a*hy2
      b(1)=1.0/hysum*hy
      b(2)=a/hysum*hy
      eyg=(ey1*b(1)+ey2*b(2))/ey
      go to 820
c
c   compute b, isotropic case (choose correctly polarized solution).
500 abpar1=1.0-rbar11*r(1,1)
      abperp=1.0-rbar22*r(2,2)
      temp a=cdabs(abperp)
      temp b=cdabs(abpar1)
      temp=temp a/temp b
      if(temp .lt. 1.0d0) go to 600
      b(1)=1.0/hy1*hy
      b(2)=0.0
      eyg=0.0
      go to 700
600 b(1)=0.0
      b(2)=1.0/ey2*ey
      hyg=0.0
700 if(temp .lt. 10.0d0) go to 800
      if(temp .gt. 0.1d0) go to 800
      if(temp b .gt. 0.1d0) go to 800

```

```

        if(temp a .gt. 0.1d0) go to 800
        if(idbg .eq. 0) go to 820
800 print 900, abpar1, abperp
c
820 continue
    if(idbg.ge.2) print 905, b
    return
c
                                entry f fct
    entry f fct(pp,c,s,f)
    call r mtrx(pp,c,r)
    call rbars(c,s,rbar11,rbar22,ey,hy)
    if(idbg .gt. 1) print 904, r, rbar11, rbar22
c
c compute modal eqn. value
    a=(1.0-r(1,1)*rbar11)
    a=a*(1.0-r(2,2)*rbar22)
    f=a-r(1,2)*r(2,1)*rbar11*rbar22
    print 901,f
    return
c
900 format('0polarization values'/
    $ ' abpar1=',(1pe15.5,1pe13.5),' abperp=',(1pe15.5,1pe13.5))
901 format('0modal eqn value=',(1pe15.5,1pe13.5))
902 format('0p values on entry to wf bndy'/
    $      4(1pe15.5,1pe13.5)/4(1pe15.5,1pe13.5))
904 format('0    r=',2(1pe15.5,1pe13.5)/6x,2(1pe15.5,1pe13.5)/
    $      '0rbar=',2(1pe15.5,1pe13.5))
905 format('0solution combination factors'/
    $ ' b1=',1pe15.5,1pe13.5,' b2=',1pe15.5,1pe13.5)
    end

```

```

      subroutine itrates
c
c   itrates is the control routine for finding an angle, theta, which
c   satisfies the modal equation.
c
      implicit real*8 (a-h,o-z)
      common/wf inpt/theta,freq,azimuth,codip,magfld,ceffnu(5),expnu(5),
$      topht,lwstht,delht,h,alpha,sigma,epsln,nglect
      common/itrates/dtheta,bnd rl,bnd im,maxitr,itr
      common/p mtx/p(16),pi(16)
      common/cs/c,s,ci,si
      real*8 magfld,lwstht
      complex*16 theta,c,s,ci,si,dtheta,i,f,f0,dfdt,del t
      data i/(0.d0,1.d0)/
c
      nr iter=0
      if(itr .eq. 1) then
11    call wf intg(topht,lwstht,delht,1)
        call f fct(pi,ci,si,f0)
        call f fct(p,c,s,f)
        dfdt=(f-f0)/dtheta
        del t=-f/dfdt
        theta=theta+del t
        print 900,theta
        nr iter=nr iter+1
        if(nr iter .gt. maxitr) then
          print *, 'max itr exceeded'
          stop
        end if
c
        del rl=del t
        if(dabs(del rl) .gt. bnd rl) go to 11
        del im=-i*del t
        if(dabs(del im) .gt. bnd im) go to 11
      end if
c
      call wf intg(topht,lwstht,delht,0)
      call f fct(p,c,s,f)
      return
c
900  format('0new theta=',2f10.3)
      end

```

```

      subroutine p deriv(p,dpdh)
c
c  p deriv computes the height derivatives of the field vectors,
c  p, according to clemmow and heading (1954).
c  equation is  $dp/dz = -ik \cdot t \cdot p$ .
c  multiplication by -i is performed by operating on real and imag parts.
c  multiplication by k is performed in routine wf step.
c
      common/t mtx/t11,t31,t41,t12,t32,t42,t14,t34,t44
      complex*16 zp(4),deriv,
$      t11,t31,t41,t12,t32,t42,t14,t34,t44
      real*8 p(8,2),dpdh(8,2),part(2)
      equivalence (deriv,part)
c
      do 11 j=1,2
      zp(1)=dcmplx(p(1,j),p(2,j))
      zp(2)=dcmplx(p(3,j),p(4,j))
      zp(3)=dcmplx(p(5,j),p(6,j))
      zp(4)=dcmplx(p(7,j),p(8,j))
      deriv=t11*zp(1)+t12*zp(2)+t14*zp(4)
      dpdh(1,j)= part(2)
      dpdh(2,j)=-part(1)
      deriv=zp(3)
      dpdh(3,j)= part(2)
      dpdh(4,j)=-part(1)
      deriv=t31*zp(1)+t32*zp(2)+t34*zp(4)
      dpdh(5,j)= part(2)
      dpdh(6,j)=-part(1)
      deriv=t41*zp(1)+t42*zp(2)+t44*zp(4)
      dpdh(7,j)= part(2)
11  dpdh(8,j)=-part(1)
      return
      end

```

```

function cdang(arg)
implicit real*8 (a-h,o-z)
complex*16 arg,mi/(0.d0,-1.d0)/
c
  argr=arg
  argi=arg*mi
  cdang=datan2(argi,argr)
  return
end

```

APPENDIX B

SOURCE LISTING FOR PROGRAM WFPLTS

A BRIEF DESCRIPTION OF USAGE OF PROGRAM WFPLTS

If the namelist logical variable SAVPLT is equal to .TRUE. (which is the default value), data are written to unit 10 for optional further processing and plotting by the program WFPLTS. WFPLTS has two sources of input: the data file generated by WAVFLD and namelist input. The program prompts the user for the name of the WAVFLD data file and then asks for the namelist data. There are four variables in the namelist: the size of the x-axis, the size of the y-axis in inches, a scaling factor and a plotting flag. The plotting flag is NPLOT, an array of six integers. It controls which plots are generated. A maximum of six plots may be generated.

If NPLOT(1) is non-zero Q and ATTEN are plotted.

If NPLOT(2) is non-zero the magnitude of SX, SY and SZ are plotted.

If NPLOT(3) is non-zero angles between SX and SZ, SY and SZ, B DOT S are plotted.

If NPLOT(4) is non-zero X, Y, and Z are plotted.

If NPLOT(5) is non-zero the magnitude of EX, EY, and EZ are plotted.

If NPLOT(6) is non-zero the magnitude of HX, HY, and HZ are plotted.

When the program prompts the user for the namelist input, default values for all variables are listed, so only variables the user wishes to change need be entered.


```

    plotx(j,2)=alog10(amax1(1.e-10,attn-plotx(j,2)))
    do 101 k=1,2
101  if(amax .lt. plotx(j,k)) amax=plotx(j,k)
    xmax=aint(amax/scalex+.99)*scalex
    xmin=xmax-size*scalex
    xscale=scalex
    call border(size,xmin,xmax,xscale,1, sizey,ymin,ymax,10.,1)
c
    do 102 j=1,nrht
    do 102 k=1,2
102  if(plotx(j,k) .lt. xmin) plotx(j,k)=xmin
    call symbol(xp,yp,.1,'q',0.,2)
    xp=xp+0.25
    call curve(xl,yl,ul,2,-xp,-yp,1.,1.,1)
    xp=xp+1.2
    call curve(plotx(1,1),ploty,up,nrht,xmin,ymin,xscale,yscale,1)
    call newpen(2)
    call symbol(xp,yp,.1,'at',0.,2)
    xp=xp+0.25
    call curve(xl,yl,ul,2,-xp,-yp,1.,1.,4)
    xp=xp+1.2
    call curve(plotx(1,2),ploty,up,nrht,xmin,ymin,xscale,yscale,4)
    go to 298
c
103  amax=-1000.
    do 104 j=1,nrht
    do 104 k=3,5
104  if(amax .lt. plotx(j,k)) amax=plotx(j,k)
    xmax=aint(amax/scalex+.99)*scalex
    xmin=xmax-size*scalex
    xscale=scalex
    call border(size,xmin,xmax,xscale,1, sizey,ymin,ymax,10.,1)
c
    do 105 j=1,nrht
    do 105 k=3,5
105  if(plotx(j,k) .lt. xmin) plotx(j,k)=xmin
    call symbol(xp,yp,.1,'sx',0.,2)
    xp=xp+0.25
    call curve(xl,yl,ul,2,-xp,-yp,1.,1.,1)
    xp=xp+1.2
    call curve(plotx(1,3),ploty,up,nrht,xmin,ymin,xscale,yscale,1)
    call newpen(2)
    call symbol(xp,yp,.1,'sy',0.,2)
    xp=xp+0.25
    call curve(xl,yl,ul,2,-xp,-yp,1.,1.,2)
    xp=xp+1.2
    call curve(plotx(1,4),ploty,up,nrht,xmin,ymin,xscale,yscale,2)
    call newpen(3)
    call symbol(xp,yp,.1,'sz',0.,2)
    xp=xp+0.25
    call curve(xl,yl,ul,2,-xp,-yp,1.,1.,4)
    xp=xp+1.2
    call curve(plotx(1,5),ploty,up,nrht,xmin,ymin,xscale,yscale,4)
    go to 298
c
106  xmin=-180.
    xmax= 180.
    xscale=360./size

```

```

call border(size,xmin,xmax,xscale,1, sizey,ymin,ymax,10.,1)
call symbol(xp,yp,.1,'atan(sz/sx)',0.,11)
xp=xp+1.15
call curve(xl,yl,ul,2,-xp,-yp,1.,1.,1)
xp=xp+1.2
call curve(plotx(1,6),ploty,up,nrht,xmin,ymin,xscale,yscale,1)
call newpen(2)
call symbol(xp,yp,.1,'atan(sz/sy)',0.,11)
xp=xp+1.15
call curve(xl,yl,ul,2,-xp,-yp,1.,1.,2)
xp=xp+1.2
call curve(plotx(1,7),ploty,up,nrht,xmin,ymin,xscale,yscale,2)
call newpen(3)
call symbol(xp,yp,.1,'acos(b . s)',0.,11)
xp=xp+1.15
call curve(xl,yl,ul,2,-xp,-yp,1.,1.,4)
xp=xp+1.2
call curve(plotx(1,8),ploty,up,nrht,xmin,ymin,xscale,yscale,4)
go to 298

c
110  amax=-1000.
      do 111 j=1,nrht
      do 111 k=9,10
111  if(amax .lt. plotx(j,k)) amax=plotx(j,k)
      xmax=aint(amax/scalex+.99)*scalex
      xmin=xmax-size*xscale
      xscale=scalex
      call border(size,xmin,xmax,xscale,1, sizey,ymin,ymax,10.,1)
      do 112 j=1,nrht
      do 112 k=9,10
112  if(plotx(j,k) .lt. xmin) plotx(j,k)=xmin
      call symbol(xp,yp,.1,'x',0.,1)
      xp=xp+0.15
      call curve(xl,yl,ul,2,-xp,-yp,1.,1.,5)
      xp=xp+1.2
      call curve(plotx(1, 9),ploty,up,nrht,xmin,ymin,xscale,yscale,5)
      call newpen(2)
      call symbol(xp,yp,.1,'z',0.,1)
      xp=xp+0.15
      call curve(xl,yl,ul,2,-xp,-yp,1.,1.,3)
      xp=xp+1.2
      call curve(plotx(1,10),ploty,up,nrht,xmin,ymin,xscale,yscale,3)
      call newpen(3)
      call symbol(xp,yp,.1,'y',0.,1)
      xp=xp+0.15
      call curve(xl,yl,ul,2,-xp,-yp,1.,1.,1)
      xp=(alog10(capy)-xmin)/xscale
      if(xp .le. 0.) go to 298
      if(xp .ge. size) go to 298
      call plot(xp, 0.,3)
      call plot(xp,sizey,2)
      go to 298

c
115  amax=-1000.
      do 116 j=1,nrht
      do 116 k=11,13
116  if(amax .lt. plotx(j,k)) amax=plotx(j,k)
      xmax=aint(amax/scalex+.99)*scalex

```

```

xmin=xmax-size*xscale
xscale=scalex
call border(size,xmin,xmax,xscale,1, sizey,ymin,ymax,10.,1)
do 117 j=1,nrht
dc 117 k=11,13
117 if(plotx(j,k) .lt. xmin) plotx(j,k)=xmin
call symbol(xp,yp,.1,'ex',0.,2)
xp=xp+0.25
call curve(xl,yl,ul,2,-xp,-yp,1.,1.,1)
xp=xp+1.2
call curve(plotx(1,11),ploty,up,nrht,xmin,ymin,xscale,yscale,1)
call newpen(2)
call symbol(xp,yp,.1,'ey',0.,2)
xp=xp+0.25
call curve(xl,yl,ul,2,-xp,-yp,1.,1.,3)
xp=xp+1.2
call curve(plotx(1,12),ploty,up,nrht,xmin,ymin,xscale,yscale,3)
call newpen(3)
call symbol(xp,yp,.1,'ez',0.,2)
xp=xp+0.25
call curve(xl,yl,ul,2,-xp,-yp,1.,1.,4)
xp=xp+1.2
call curve(plotx(1,13),ploty,up,nrht,xmin,ymin,xscale,yscale,4)
go to 298
c
120 amax=-1000.
do 121 j=1,nrht
do 121 k=14,16
121 if(amax .lt. plotx(j,k)) amax=plotx(j,k)
xmax=aint(amax/scalex+.99)*scalex
xmin=xmax-size*xscale
xscale=scalex
call border(size,xmin,xmax,xscale,1, sizey,ymin,ymax,10.,1)
do 122 j=1,nrht
do 122 k=14,16
122 if(plotx(j,k) .lt. xmin) plotx(j,k)=xmin
call symbol(xp,yp,.1,'hx',0.,2)
xp=xp+0.25
call curve(xl,yl,ul,2,-xp,-yp,1.,1.,1)
xp=xp+1.2
call curve(plotx(1,14),ploty,up,nrht,xmin,ymin,xscale,yscale,1)
call newpen(2)
call symbol(xp,yp,.1,'hy',0.,2)
xp=xp+0.25
call curve(xl,yl,ul,2,-xp,-yp,1.,1.,3)
xp=xp+1.2
call curve(plotx(1,15),ploty,up,nrht,xmin,ymin,xscale,yscale,3)
call newpen(3)
call symbol(xp,yp,.1,'hz',0.,2)
xp=xp+0.25
call curve(xl,yl,ul,2,-xp,-yp,1.,1.,4)
xp=xp+1.2
call curve(plotx(1,16),ploty,up,nrht,xmin,ymin,xscale,yscale,4)
c
298 call pltend
299 continue
go to 90
c

```

```
999  stop
1000 format(a40)
2000 format('f=',f7.3,' r=',f7.3,' a=',f7.3,' c=',f7.3,' s=',1pe7.1)
      end
```

```

      subroutine curve(x,y,up,nrpts,xmin,ymin,xinc,yinc,line)
c
c x,y,up must be dimensioned at least nrpts
c xmin,ymin are x,y origin in user units
c xinc,yinc are x,y scales in user units per inch
c
c line=1:  solid
c       2:  long dash
c       3:  medium dash
c       4:  short dash
c       5:  dotted
c       6:  short + long dash
c       7:  short + short + long dash
c
      logical up,up1,up2
      dimension ipen(10),joc(7),x(nrpts),y(nrpts),up(nrpts)
      data ipen/3,2,3,2,3,2,2,2,2,2/,joc/18, 61, 56, 54, 52, 11, 36/
      data delr/.1/
c
      if(nrpts .le. 1) go to 99
c
      if(line) 1,2,3
1      kk=mod(line,7)+7
      go to 4
2      kk=0
      go to 4
3      kk=mod(line,7)
4      kk=kk+1
      jo=joc(kk)/10
      jc=joc(kk)-10*jo
c
      j=1
      ip=2
      if(kk .eq. 6) ip=3
      dr=0.
      rho1=0.
      rho2=delr
      px1=(x(1)-xmin)/xinc
      py1=(y(1)-ymin)/yinc
      up1=up(1)
      if(up1) go to 10
c
c go to first position with pen up
      call plot(px1,py1,3)
c
10     do 40 i=2,nrpts
          px2=(x(i)-xmin)/xinc
          py2=(y(i)-ymin)/yinc
          up2=up(i)
          if(up2) go to 22
          if(up1) go to 37
          if(kk .eq. 2) go to 38
          delx=px2-px1
          dely=py2-py1
          rho=sqrt(delx**2+dely**2)
          rho1=rho1+rho
          if(rho2 .gt. rho1) go to 38
          delx=delx*delr/rho

```

```

      dely=dely*delr/rho
      dx 6=delx*.1
      dy 6=dely*.1
      if(dr .eq. 0.) go to 20
      dx=delx*dr/delr
      dy=dely*dr/delr
      px1=px1+dx
      pyl=pyl+dy
      go to 21
20    if(rho2 .gt. rho1) go to 38
      px1=px1+delx
      pyl=pyl+dely
21    call plot(px1,pyl,ip)
      if(kk .eq. 6) call plot(px1+dx6,pyl+dy6,2)
      j=j+1
      ip=ipen(jo+mod(j,jc))
      rho2=rho2+delr
      go to 20
22    dr=0.
      rho1=0.
      rho2=delr
      go to 39
c pen has been up, prepare to lower pen
37    call plot(px2,py2,3)
      go to 39
38    call plot(px2,py2,ip)
      dr=rho2-rho1
39    px1=px2
      pyl=py2
      up1=up2
40    continue
99    return
      end

```

```

subroutine border(xlng,xmin,xmax,xinc,nx,ylng,ymin,ymax,yinc,ny)
c
dimension xinc(nx),yinc(ny)
logical fy,fx
c
fx=.false.
fy=.false.
if(nx .eq. 1) fx=.true.
if(ny .eq. 1) fy=.true.
xt=xlng-.1
yt=ylng-.1
xscale=xlng/(xmax-xmin)
yscale=ylng/(ymax-ymin)
ym=abs(ymin)
yln=-.4
if(ym .ge. 10.) yln=yln-.1
if(ym .ge. 100.) yln=yln-.1
if(ym .ge. 1000.) yln=yln-.1
if(ymin .lt. 0.) yln=yln-.1
ym=abs(ymax)
ylm=-.4
if(ym .ge. 10.) ylm=ylm-.1
if(ym .ge. 100.) ylm=ylm-.1
if(ym .ge. 1000.) ylm=ylm-.1
if(ymax .lt. 0.) ylm=ylm-.1
xm=abs(xmax)
xlm=-.3
if(xm .ge. 10.) xlm=xlm-.1
if(xm .ge. 100.) xlm=xlm-.1
if(xm .ge. 1000.) xlm=xlm-.1
if(xmax .lt. 0.) xlm=xlm-.1
if(fx) dx=xinc(1)
if(fy) dy=yinc(1)
iy=1
yl=0.
call number(yln,0.,.1,ymin,0.,1)
call plot(0.,0.,3)
if(fy) go to 110
10 yp=(yinc(iy)-ymin)*yscale
go to 111
110 yl=yl+dy
yp=yl*yscale
111 if(yp .lt. 0.) go to 99
if(yp .ge. ylng) go to 11
call plot(0.,yp,2)
call plot(.1,yp,2)
call plot(0.,yp,2)
if(fy) go to 110
iy=iy+1
if(iy .le. ny) go to 10
11 call plot(0.,ylng,2)
call number(ylm,ylng-.1,.1,ymax,0.,1)
call plot(0.,ylng,3)
ix=1
xl=0.
if(fx) go to 112
12 xp=(xinc(ix)-xmin)*xscale
go to 120

```

```

112  xl=xl+dx
      xp=xl*xscale
120  if(xp .lt. 0.) go to 99
      if(xp .ge. xlng) go to 13
      call plot(xp,ylng,2)
      call plot(xp,yt,2)
      call plot(xp,ylng,2)
      if(fx) go to 112
      ix=ix+1
      if(ix .le. nx) go to 12
13   call plot(xlng,ylng,2)
      if(fy) go to 130
113  iy=iy-1
      if(iy .le. 0) go to 15
      yp=(yinc(iy)-ymin)*yscale
      go to 14
130  yl=yl-dy
      yp=yl*yscale
      if(yp .le. 0.) go to 15
14   call plot(xlng,yp,2)
      call plot(xt,yp,2)
      call plot(xlng,yp,2)
      if(fy) go to 130
      go to 113
15   call plot(xlng,0.,2)
      call number(xlng+xlm,-.2,.1,xmax,0.,1)
      call plot(xlng,0.,3)
      if(fx) go to 150
115  ix=ix-1
      if(ix .le. 0) go to 17
      xp=(xinc(ix)-xmin)*xscale
      go to 16
150  xl=xl-dx
      xp=xl*xscale
      if(xp .le. 0.) go to 17
16   call plot(xp,0.,2)
      call plot(xp,.1,2)
      call plot(xp,0.,2)
      if(fx) go to 150
      go to 115
17   call plot(0.,0.,2)
      call number(0.,-.2,.1,xmin,0.,1)
      return
99   print 100,xlng,xmin,xmax,xinc(1),nx,ylng,ymin,ymax,yinc(1),ny
100  format('0*** Error in BORDER: xlng, xmin, xmax, xinc(1), nx =',
$      'lp4e15.5,i5/24x,ylng, ymin, ymax, yinc(1), ny =',lp4e15.5,
$      'i5/'0***')
      call pltend
      stop
      end

```

END

DATE

FILMED

5-88

DTIC